Project Title

Customer Churn Analysis Using Machine Learning on the Telco Dataset

Table of Contents

1. Introduction 1
2. Dataset Selection and Justification.. 1
3. Data Preprocessing.. 2
4. Data Analysis and Visualization... 4
5. Data Mining and Machine Learning.. 8
6. Interpretation and Business Intelligence..................**Error! Bookmark not defined.**
7. Conclusion 16
8. Tools & Technologies Used 16
9. Team Contributions 17
References.. 17
Appendix 17

1. Introduction

In today’s competitive telecom industry, retaining customers is crucial for profitability. One of the key challenges telecom companies face is customer churn—when existing customers discontinue their services. This project aims to analyze real-world telecom customer data to uncover patterns and factors that contribute to churn.

The purpose of this project is to apply data analytics and machine learning techniques to predict churn and provide actionable insights. By identifying customers at high risk of leaving, businesses can develop targeted retention strategies.

Through this dataset, we aim to:

- Clean and preprocess the data for accuracy and quality
- Visualize key patterns and customer behavior
- Build ML models to predict customer churn
- Deliver insights for business intelligence and decision-making

2. Dataset Selection and Justification

- **Dataset Name**: Telco Customer Churn Dataset
- **Dataset Link**: https://www.kaggle.com/datasets/blastchar/telco-customer-churn
- **Source**: Kaggle
- **Domain**: Telecom / Business Intelligence
- **Dataset Size**: 7,043 rows × 21 columns
- **Variables**: Mix of demographic, service-related, and billing-related features including:
    - gender, SeniorCitizen, Partner, Dependents, tenure, MonthlyCharges, TotalCharges, Contract, Churn, etc.

Why This Dataset?

- **Relevance**: Directly addresses a business-critical issue—customer churn
- **Richness**: Offers categorical, numerical, and binary variables for varied analysis
- **Suitability**: Excellent for classification, clustering, visualization, and storytelling

3. Data Preprocessing

Efficient preprocessing is essential for ensuring high-quality input to data mining and machine learning algorithms. The Telco Customer Churn dataset was cleaned, transformed, and explored to prepare it for model development and insight generation.

3.1 Data Cleaning

The following steps were performed to clean and structure the data:

- **Missing Values**:

  The TotalCharges column contained 11 missing entries, resulting from blank string values. These were first converted to numeric and then imputed with the median value (1,397.47) to minimize distortion due to skewness.

- **Duplicate Removal**:

  A check for duplicate rows was conducted and all duplicate entries were removed, ensuring each customer record was unique.

- **Outlier Detection**:

  Outliers in tenure, MonthlyCharges, and TotalCharges were detected using box plots. These were retained, as they may represent important customer behavior patterns (e.g., very long-term or high-value customers).

- **Normalization**:
  Continuous variables—tenure, MonthlyCharges, and TotalCharges—were normalized using Min-Max scaling for clustering purposes.

- **Categorical Encoding**:
  - Label Encoding was applied to binary categorical variables such as gender, Partner, Dependents, PhoneService, and PaperlessBilling.
  - One-Hot Encoding was applied to multi-category variables such as Contract, PaymentMethod, and InternetService to facilitate ML model training.

3.2 Descriptive Stats

Table 3.1 presents descriptive statistics highlighting customer tenure and billing patterns. Most customers are on month-to-month contracts, with average tenure around 32 months and moderate billing levels.

Table 3.1: Descriptive Stats

Statistic	**Value**
Mean Tenure	32.37 months

Median Tenure	29 months
Minimum Tenure	0 months
Maximum Tenure	72 months
Average Monthly Charges	$64.76
Highest Monthly Charge	$118.75
Average Total Charges	$2,278.06
Most Common Contract Type	Month-to-Month

3.4 Correlation Analysis

A correlation matrix was computed to explore linear relationships among key numerical features: tenure, MonthlyCharges, and TotalCharges.

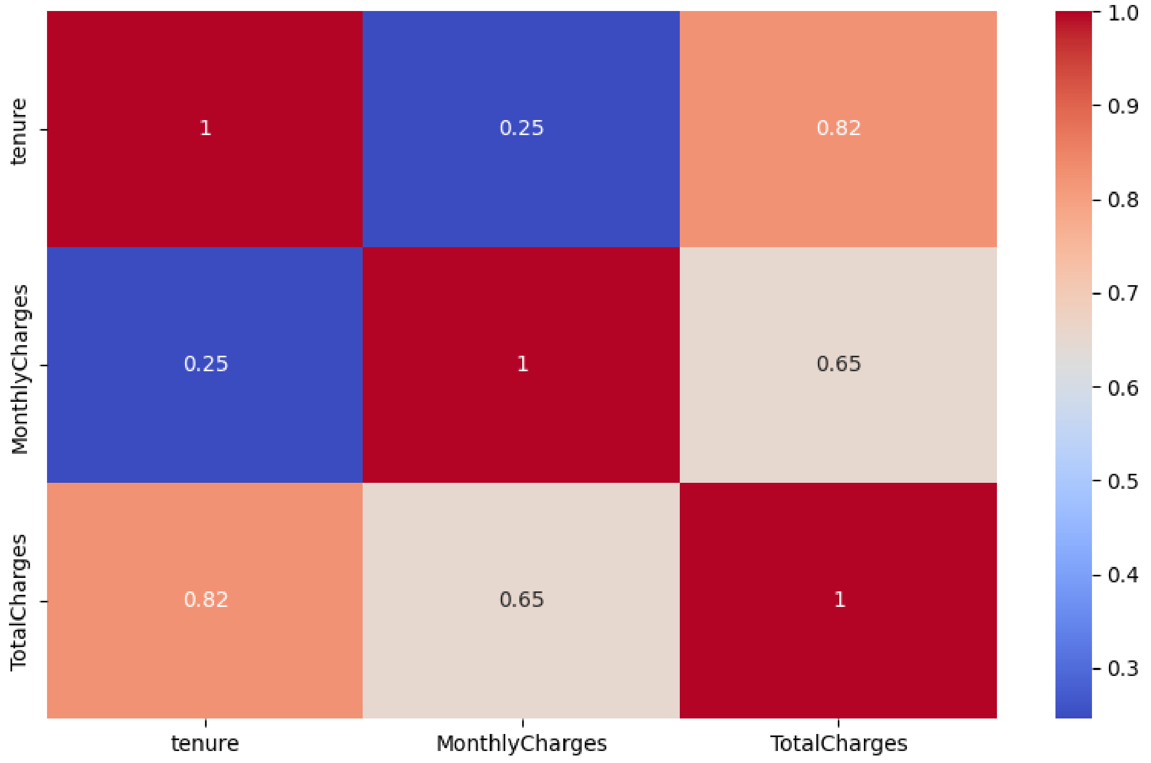


Figure 3.1: Correlation Matrix – Tenure, MonthlyCharges, and TotalCharges

- There is a strong positive correlation (0.82) between tenure and TotalCharges, as customers with longer tenure naturally accumulate more total charges.
- MonthlyCharges and TotalCharges also show a moderate positive correlation (0.65), indicating that customers with higher monthly charges tend to have higher total charges.

- Interestingly, tenure and MonthlyCharges have a weak correlation (0.25), suggesting that time spent with the company does not necessarily determine the customer's monthly expenditure.

This correlation analysis helps in understanding feature importance for modeling and provides insight into the financial behaviors of customers.

Key Trends and Insights:

- Customers with **month-to-month contracts** exhibit significantly higher churn.
- Fiber optic internet users tend to churn more than DSL users.
- Customers with longer tenure tend to have lower churn, indicating greater loyalty.
- Senior citizens and customers with electronic check payment method show relatively high churn.

These findings laid the groundwork for selecting relevant features and framing the machine learning problem in subsequent stages.

4. Data Analysis and Visualization

4. Data Analysis and Visualization

This section presents key visualizations used to explore customer behavior, spending trends, and potential churn factors. All figures are properly labeled with captions and interpreted based on the observed patterns.

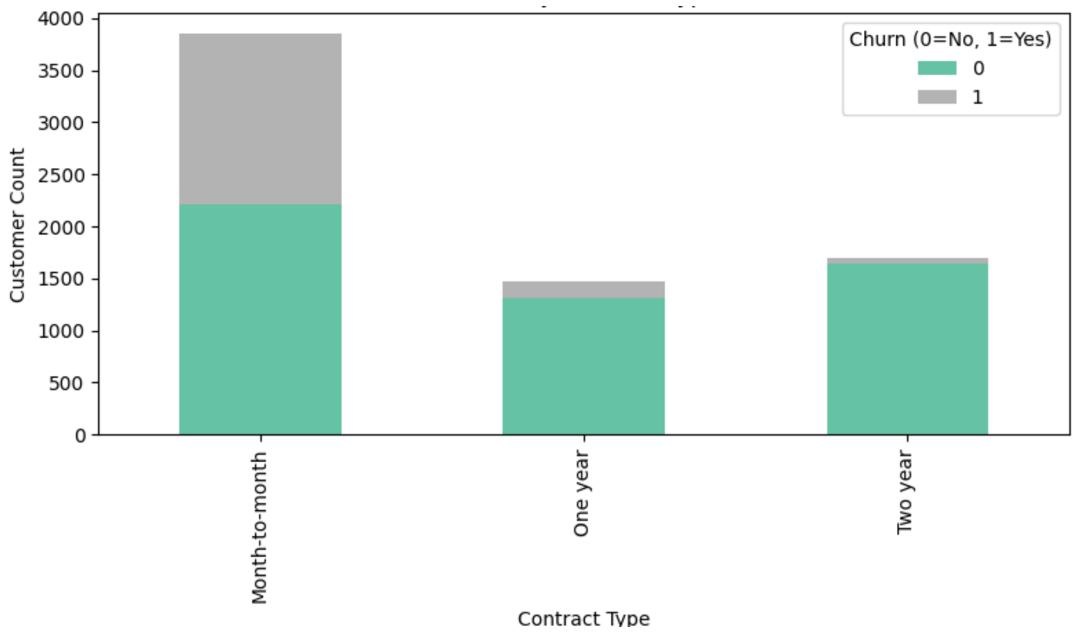


Figure 4.1: Bar Chart – Churn vs. Contract Type

This bar chart compares churn rates across different contract types (Month-to-Month, One Year, Two Year). It clearly shows that month-to-month customers have the highest churn rate, indicating instability and lower customer retention in this category.

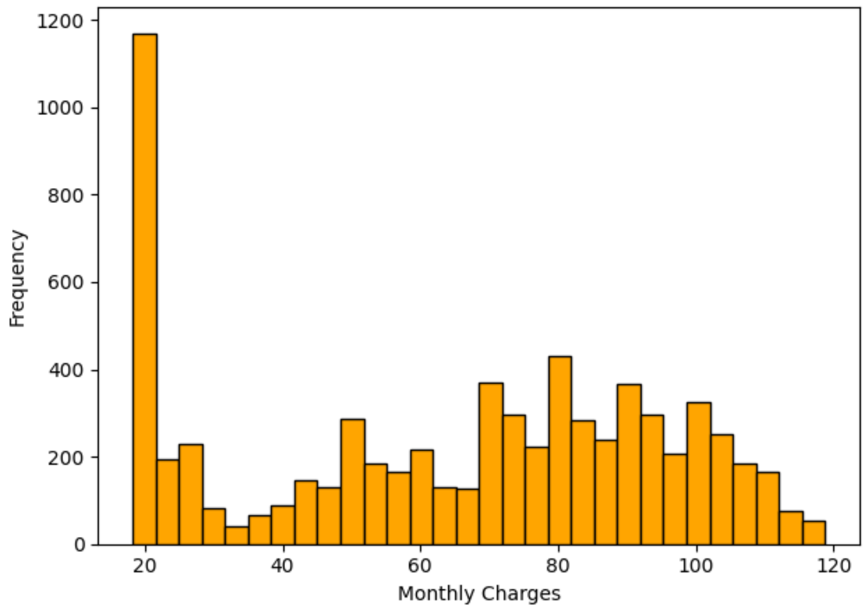


Figure 4.2: Histogram – Distribution of Monthly Charges

The histogram shows a sharp spike near $20, indicating a large group of low-paying customers, followed by a spread across higher charges up to $120, suggesting varied service levels.

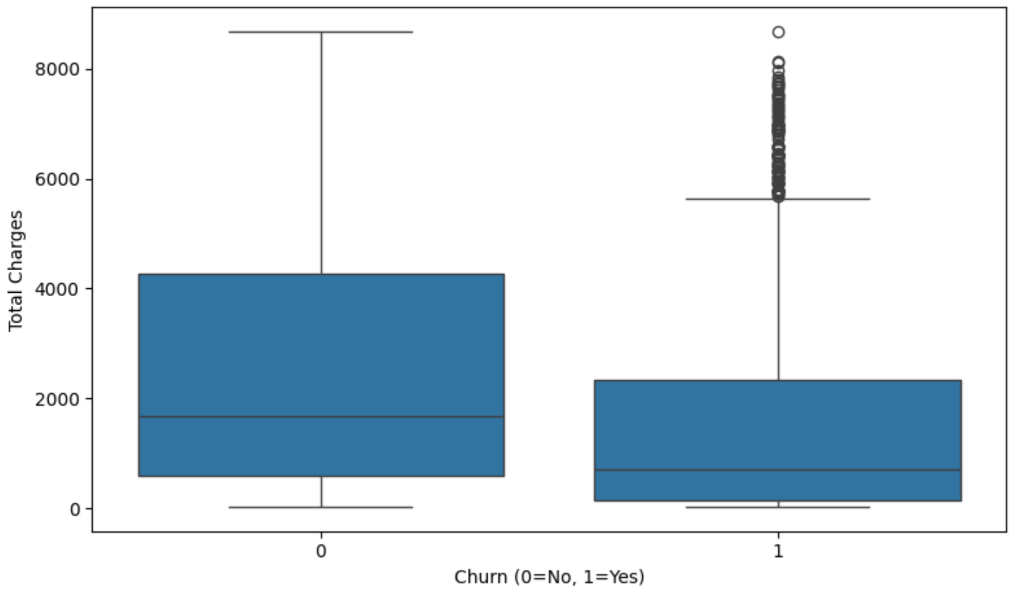


Figure 4.3: Box Plot – Total Charges by Churn Status

This box plot compares the distribution of Total Charges between customers who stayed (Churn = 0) and those who left (Churn = 1). It is evident that churned customers generally incurred lower total charges, indicating shorter tenure or lower engagement. Customers who churned tend to have paid less overall, which could suggest early dropout or lower commitment to services.

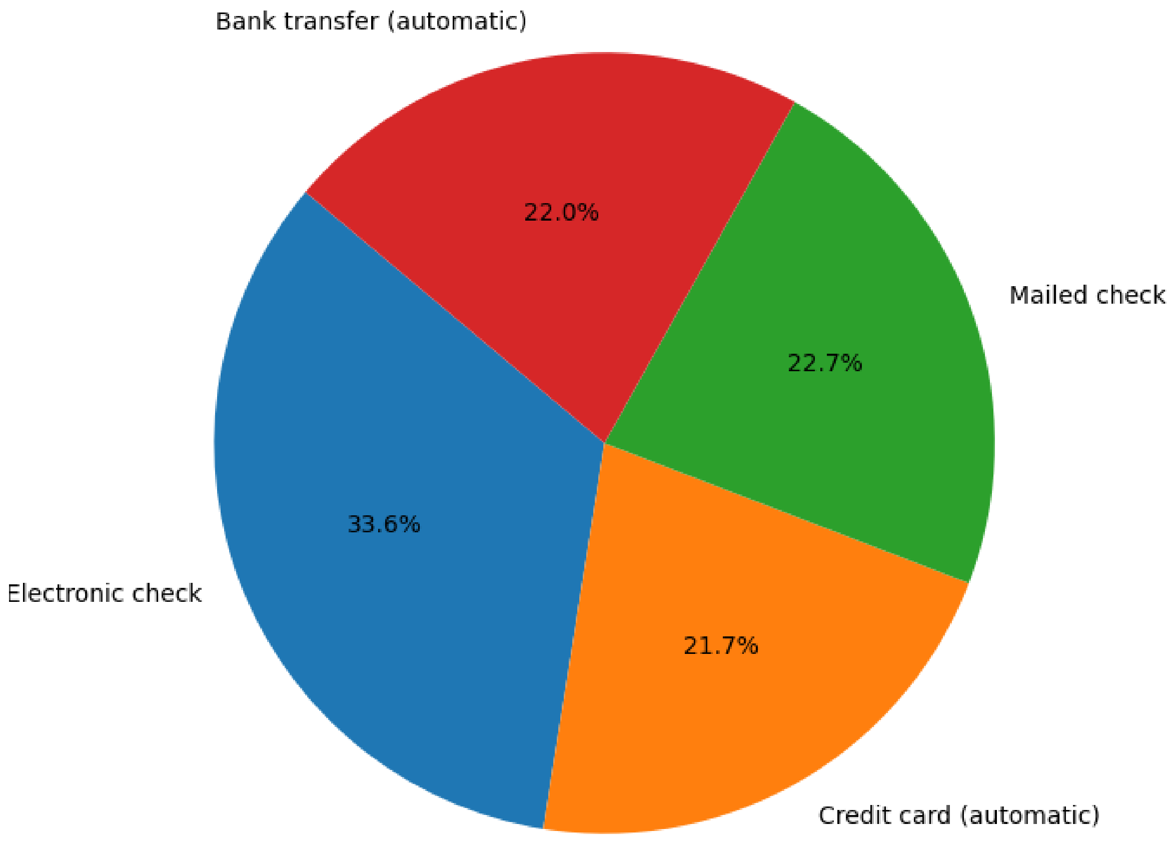


Figure 4.4: Pie Chart – Distribution of Payment Methods

This pie chart illustrates the proportions of different payment methods used by customers. The majority of customers (33.6%) prefer Electronic Check, followed by Mailed Check (22.7%), Bank Transfer (automatic) (22.0%), and Credit Card (automatic) (21.7%). The dominance of electronic checks may indicate ease of use, but this group also shows higher churn, possibly due to the absence of contract-based auto-renewal.

This line graph maps tenure against average monthly charges. A slight upward trend suggests that longer-tenured customers tend to use more services or higher-tier plans, resulting in greater monthly bills.

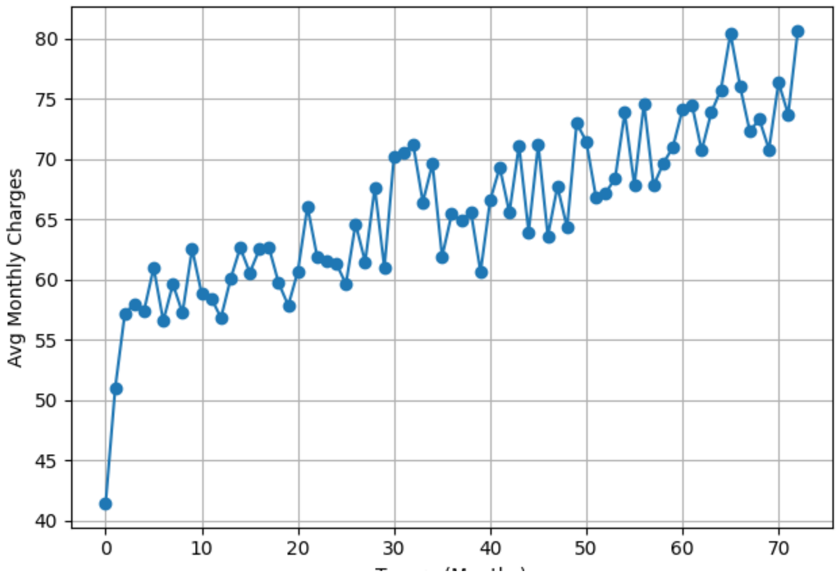


Figure 4.5: Line Graph – Tenure vs. Monthly Charges Trend

The graph indicates a gradual upward trend, showing that customers with longer tenure generally incur higher monthly charges, possibly due to service upgrades or add-ons over time.

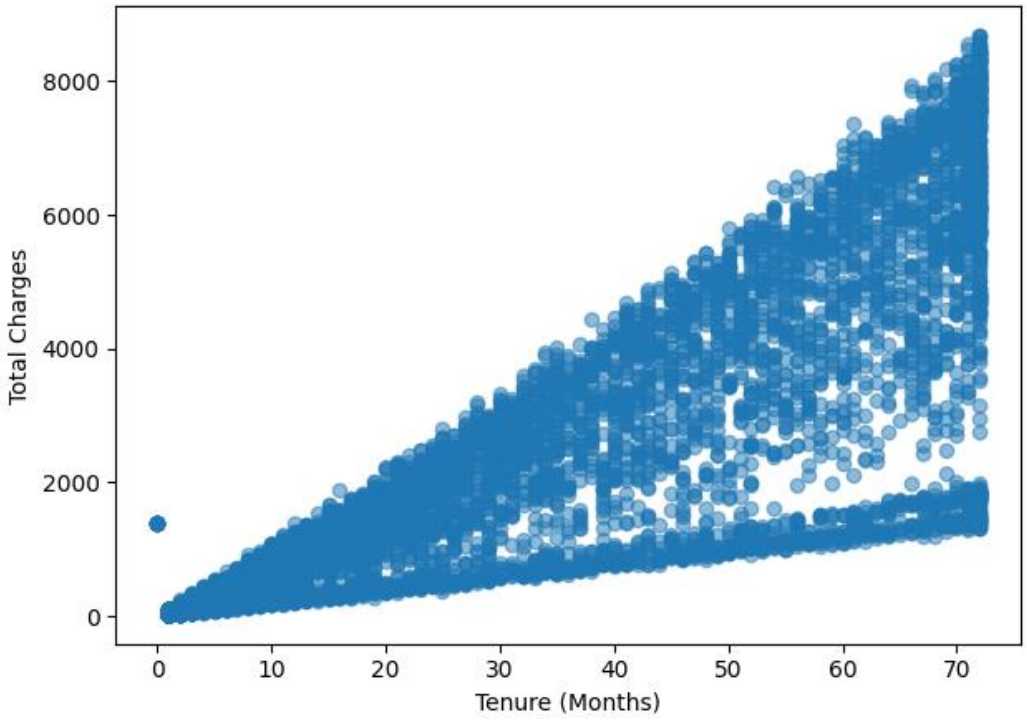


Figure 4.6: Scatter Plot – Tenure vs. Total Charges

The scatter plot shows a strong positive correlation between tenure and total charges. This is expected, as customers who stay longer accumulate more charges over time.

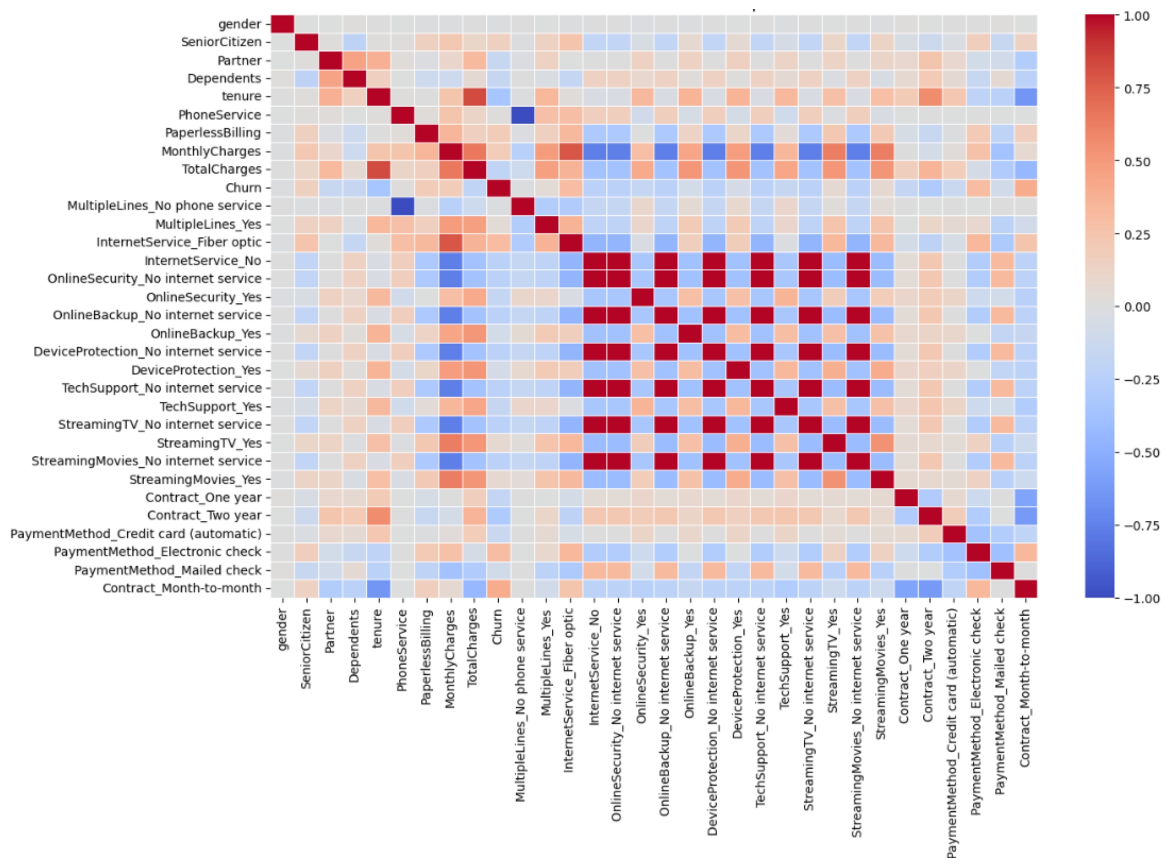


Figure 4.7: Heatmap – Correlation Matrix of Tenure, MonthlyCharges, and TotalCharges

This heatmap reveals correlations among the key numerical features. The strongest correlation is between tenure and total charges (0.82), while monthly charges also show moderate correlation with total charges (0.65).

5. Data Mining and Machine Learning

5.1 Problem Framing

- **Goal**: Predict customer churn (Yes/No)
- **Target Variable**: Churn
- **Train-Test Split**: 80% train, 20% test

5.2 Machine Learning Models Used

Table 5.1 provides an overview of the four machine learning models implemented in this study. Each model was selected based on its interpretability, robustness, or suitability for clustering tasks. Logistic Regression served as a baseline, while Random Forest addressed overfitting issues effectively. Both Python and Orange tools were used, with key metrics like accuracy and silhouette score for evaluation.

Table 5.1: Machine Learning Models Used

Model	Reason for Use	Tool	Key Metric
Logistic Regression	Simple baseline model	Python	Accuracy
Decision Tree	Easy to interpret decision flow	Orange	Accuracy
Random Forest	Robust, handles overfitting	Python	Accuracy
K-Means Clustering	To segment customers by behavior	Orange	Silhouette

5.3 Comparison Table of ML Results

Table 5.2 compares the models based on their classification performance across multiple metrics. Logistic Regression achieved the highest AUC, while Decision Tree showed better recall for churn prediction. Random Forest balanced performance but had slightly lower recall, and K-Means was excluded due to unsupervised nature.

Table 5.2: Comparison Results

Model	Accuracy	Precision (1)	Recall (1)	F1-score (1)	AUC
Logistic Regression	**0.793**	**0.60**	**0.54**	**0.57**	**0.846**
Decision Tree	**0.784**	**0.56**	**0.64**	**0.60**	**0.830**
Random Forest	**0.785**	**0.59**	**0.48**	**0.53**	**0.822**

5.4 ML Visualizations

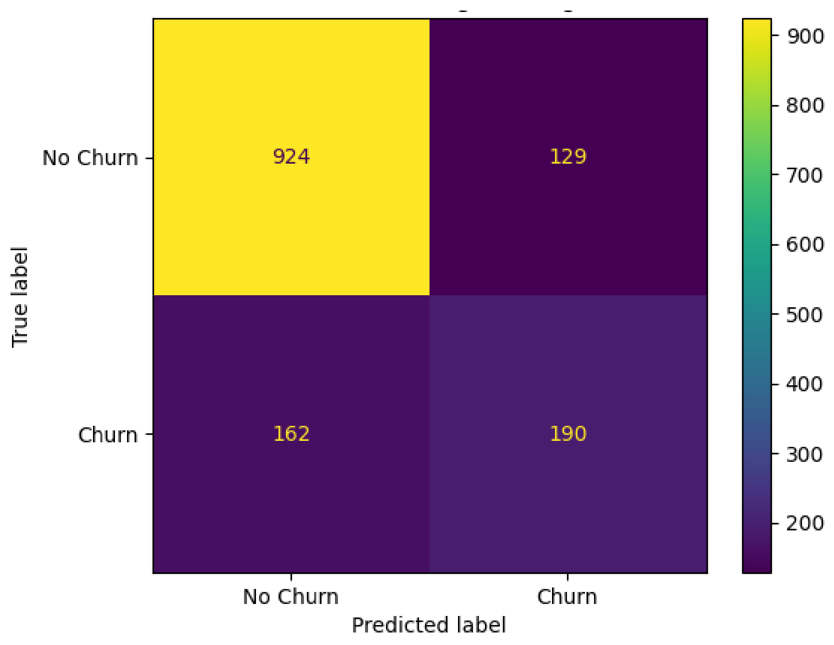


Figure 5.1: Confusion Matrix – Logistic Regression

This matrix illustrates the performance of the logistic regression model in predicting customer churn. Out of the total cases, 924 non-churned customers were correctly classified (True Negatives), while 129 were incorrectly predicted as churn (False Positives). Similarly, 190 churned customers were correctly identified (True Positives), and 162 were misclassified as non-churn (False Negatives).

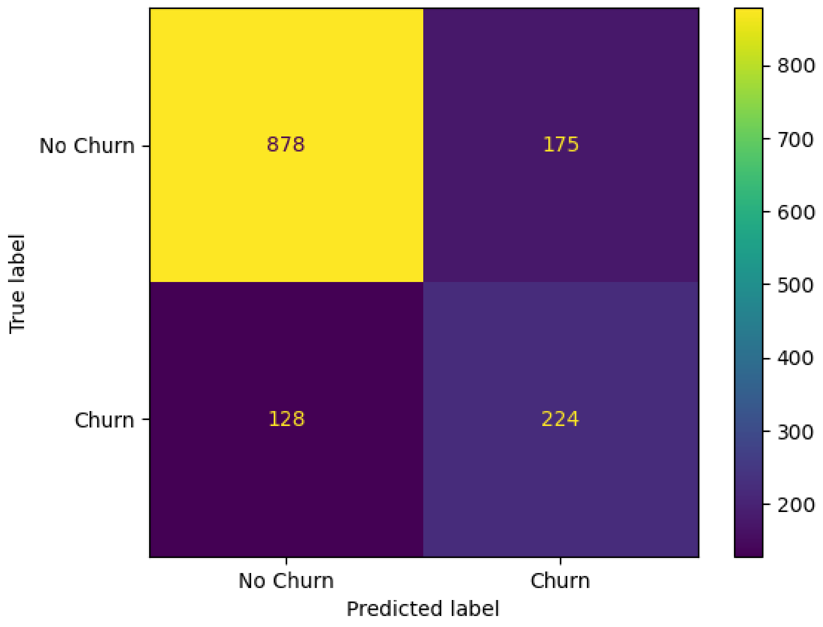


Figure 5.2: Confusion Matrix – Decision Tree

This confusion matrix, Figure 5.2, visualizes the classification performance of the Decision Tree model. It accurately identified 878 customers who did not churn (True Negatives) and correctly classified 224 churned customers (True Positives). However, 175 non-churned customers were misclassified as churn (False Positives), while 128 churned customers were incorrectly labeled as non-churn (False Negatives).

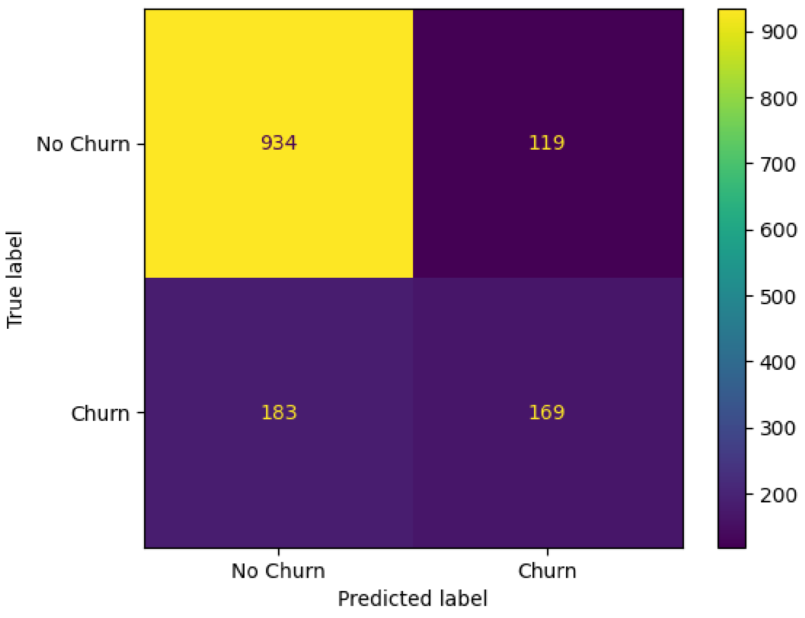


Figure 5.3: Confusion Matrix – Random Forest

This confusion matrix, Figure 5.3, displays the performance of the Random Forest classifier. The model correctly predicted 934 non-churn cases (True Negatives) and 169 churn cases (True Positives). However, it misclassified 119 non-churn customers as churn (False Positives) and 183 churned customers as non-churn (False Negatives). This indicates high precision on non-churn but room for improvement in detecting churned customers.

ROC Curve

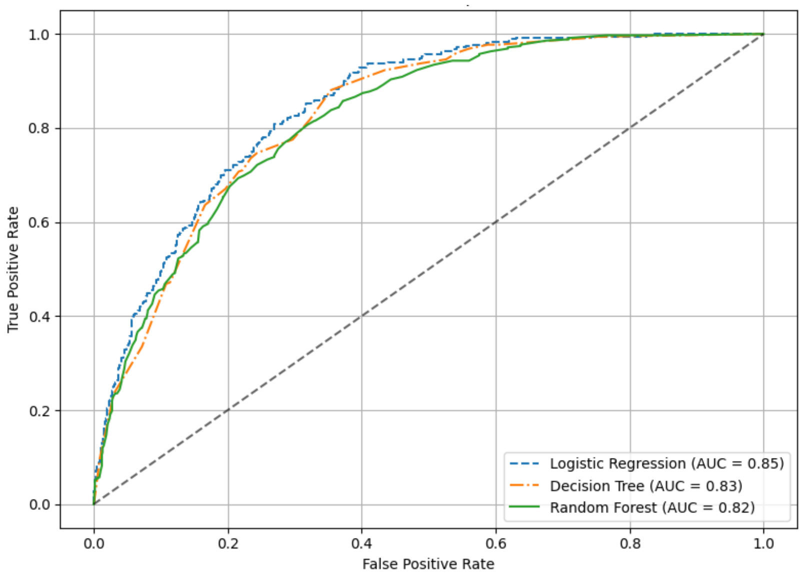


Figure 5.4: ROC Curve Comparison of Classification Models

This ROC curve compares the performance of three classifiers: Logistic Regression, Decision Tree, and Random Forest. The area under the curve (AUC) indicates the model's ability to distinguish between churn and non-churn classes. Logistic Regression achieved the highest AUC of 0.846, followed by Decision Tree (0.83) and Random Forest (0.822), suggesting that Logistic Regression offers the best overall discriminatory power among the tested models.

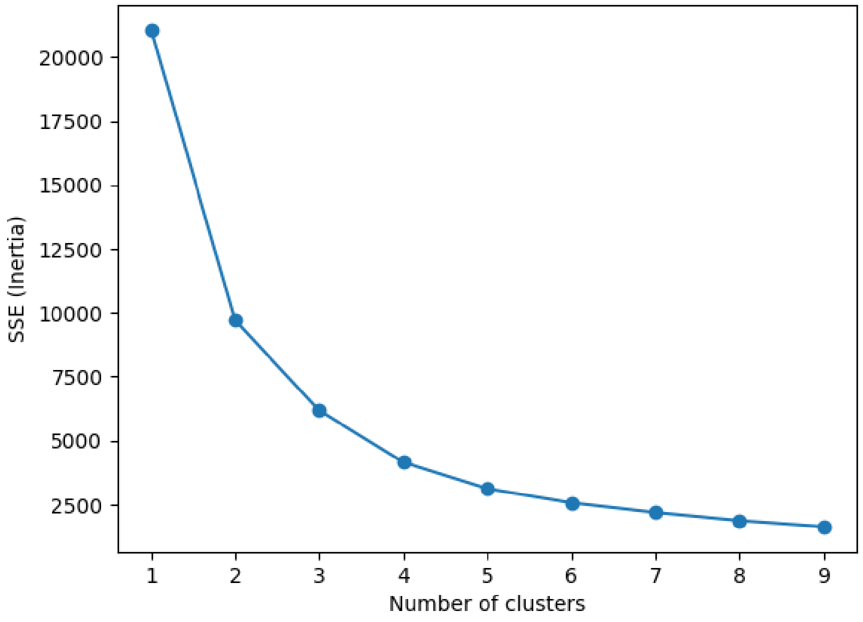


Figure 5.5: Elbow Method for Optimal k

This figure, 5.5 uses the Elbow Method to identify the optimal number of clusters for k-means. The elbow point appears at k = 5, suggesting 5 is an appropriate number of customer segments to balance between overfitting and underfitting.

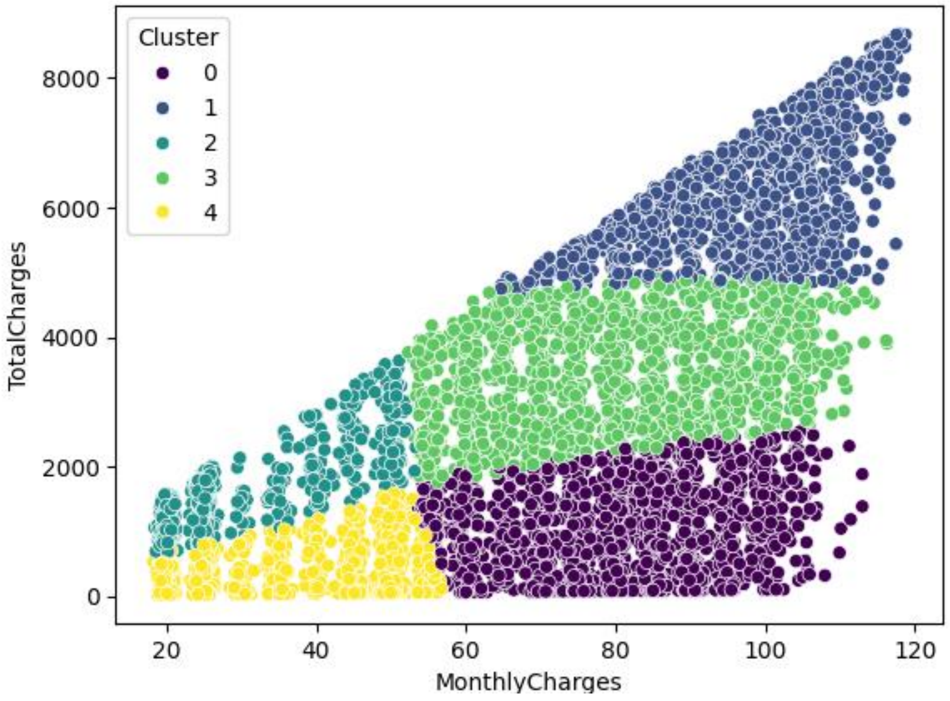


Figure 5.6: Customer Segments by Charges

This scatter plot, 5.6, shows 5 customer clusters plotted by Monthly Charges and Total Charges.Each cluster is color-coded and shows distinct customer spending patterns, indicating effective segmentation.

Table 5.3 summarizes customer segments based on tenure and spending patterns using K-Means clustering. The clusters reveal strategic insights for targeting retention, upselling, and churn reduction initiatives.

Table 5.3: Cluster Profile Summary

Cluster	Avg. Tenure (months)	Avg. Monthly Charges	Avg. Total Charges	Count (Customers)	Insights
0	11.45	79.56	907.00	1869	New customers with high monthly spending but low total charges—likely recent sign-ups.
1	64.71	97.93	6322.86	1233	Loyal high-value customers—long tenure and high spending. Important for retention.
2	54.08	27.50	1475.47	908	Long-term low spenders—stable but low revenue contributors. Potential for upselling.
3	44.77	79.87	3477.63	1358	Mid-tier customers—moderate tenure and charges. Represent growth opportunity.
4	10.21	31.72	312.84	1653	New and low-spending users—highest churn risk. May need onboarding or promotional support.

These clusters provide actionable insights for personalized marketing, churn prevention, and service tier optimization. You can now tailor strategies for each group, such as:

- Offering loyalty rewards to Cluster 1
- Cross-selling to Cluster 2 and 3
- Onboarding support or offers for Cluster 0 and 4

6. Interpretation and Business Intelligence

Key Insights

A comprehensive evaluation of machine learning models, data visualizations, and customer segmentation has revealed several critical patterns:

- **Contract Type and Churn Propensity**

  Customers on month-to-month contracts exhibit the highest churn rates, indicating lower commitment levels and a pressing need for strategies that foster long-term engagement.

- **Payment Method Influence**

  Electronic check users show significantly higher churn, suggesting potential dissatisfaction due to manual payment processes, billing issues, or lack of convenience.

- **Tenure as a Retention Indicator**

  A strong inverse relationship exists between customer tenure and churn likelihood. Long-tenured users demonstrate higher loyalty and satisfaction.

- **Increased Risk Among Senior Citizens**

  Senior customers—particularly those without fixed-term contracts—are more susceptible to churn, possibly due to usability concerns, pricing sensitivity, or insufficient support.

- **Service Type Matters**

  Fiber optic users are more likely to churn than DSL users. This may stem from performance expectations, service disruptions, or perceived value gaps.

Business Interpretation

These insights yield several strategic interpretations that can guide business action:

- **Revenue vs. Churn Correlation**

Customers with shorter tenures and lower total charges are disproportionately represented among churners. These individuals may feel less invested in the service and should be engaged proactively.

- **Customer Segmentation Insights**

  The K-Means clustering analysis categorized the customer base into five segments, each representing unique churn risk profiles:

  - Cluster 1: High-value, long-tenured customers — vital for retention efforts and loyalty programs.
  - Cluster 4: New, low-spending users — most at risk of churn and in need of onboarding, education, and incentives.
  - Cluster 2: Long-term, low-revenue customers — offer upselling opportunities to increase lifetime value.

- **Model Performance Summary**

  Among the three classification models, Logistic Regression achieved the best AUC score (0.846), effectively distinguishing between churners and non-churners. However, F1-scores indicate that churn recall remains moderate, suggesting scope for performance enhancement via ensemble methods or deep learning models.

Strategic Recommendations

Based on the analytical findings, the following recommendations are proposed to reduce churn and strengthen customer relationships:

- **Encourage Long-Term Commitments**

  Offer discounts, bundled packages, or loyalty benefits to incentivize customers to shift from monthly to annual or biennial contracts.

- **Promote Auto-Payment Methods**

Educate users on the advantages of automated payment channels (e.g., credit card, bank transfer) to reduce the churn linked to manual or electronic check methods.

- **Segment-Specific Retention Strategies**

  - Cluster 4: Implement welcome offers, guided onboarding, and regular check-ins.
  - Cluster 2: Launch upselling campaigns and reward schemes to enhance perceived value.
  - Cluster 1: Maintain engagement with loyalty rewards, personalized communications, and premium support.

- **Tailor Support for Vulnerable Groups**

  Improve support infrastructure for fiber optic customers and senior citizens by addressing service quality concerns and offering tailored assistance.

- **Operationalize Churn Prediction**

  Integrate the churn prediction model into the CRM system to enable real-time risk scoring. Use alerts to trigger automated retention workflows for at-risk customers.

7. Conclusion

This report successfully demonstrated how data mining and machine learning can be applied to solve a critical business challenge—predicting customer churn in the telecom industry. By leveraging a structured data science pipeline, we gained both predictive power and actionable insights.

Key stages included data preprocessing, exploratory data analysis, customer segmentation using K-Means clustering, and the deployment of classification models such as Logistic Regression, Decision Tree, and Random Forest. Among these, Logistic Regression achieved the highest AUC score (0.846), indicating strong discriminative ability for churn prediction.

In addition to model evaluation, business intelligence insights revealed that factors such as contract type, tenure, payment method, and service type significantly influence churn behavior. Segmenting customers into distinct clusters enabled a more targeted understanding of churn risk and revenue potential.

The findings inform a range of strategic recommendations—from promoting long-term contracts and encouraging auto-payment adoption to designing personalized retention campaigns for at-risk segments. Integrating the predictive model into CRM systems further enhances the opportunity for real-time churn prevention.

While the models provide a solid foundation, future improvements could include advanced techniques such as ensemble stacking, deep learning, or real-time analytics to enhance prediction accuracy and operational responsiveness. Overall, this project provides a robust framework for using data-driven decision-making to improve customer retention and business performance.

8. Tools & Technologies Used

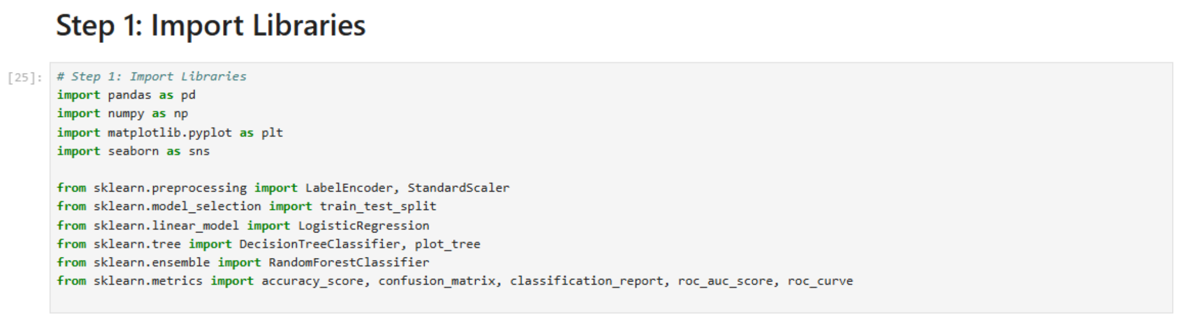
- **Tools**: Python (Jupyter Notebook), Excel
- **Libraries**: pandas, seaborn, matplotlib, scikit-learn
- **Platform**: Local Jupyter Notebook on Windows

9. Team Contributions

References

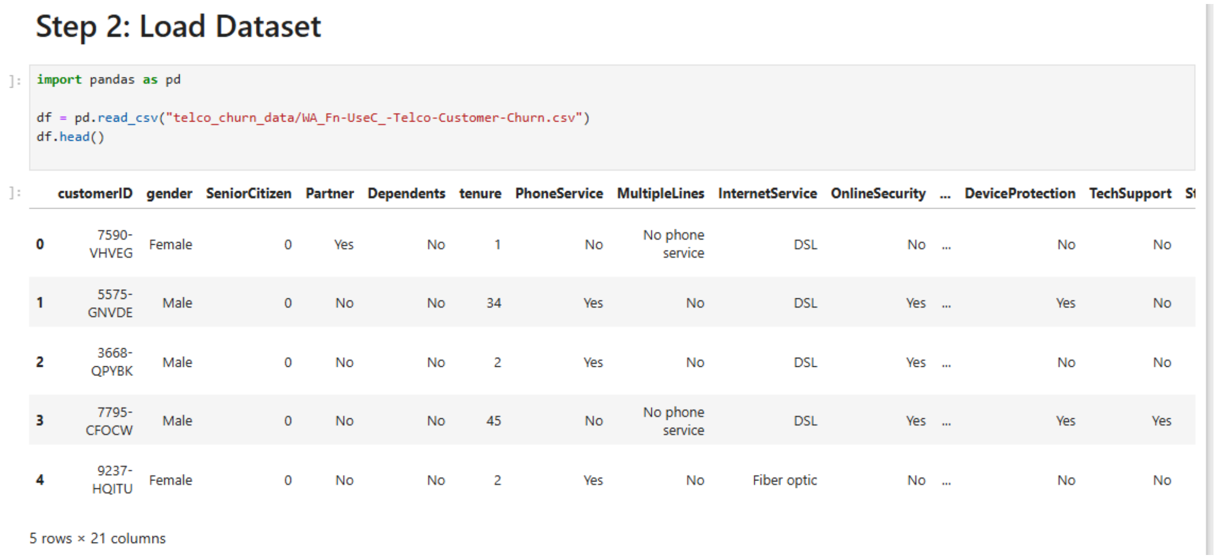
- Kaggle. (n.d.). *Telco Customer Churn*. Retrieved from https://www.kaggle.com/datasets/blastchar/telco-customer-churn
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. O'Reilly.
- Orange Data Mining. (2024). *Official Documentation*.

Appendix



Step 1: Import Libraries

```
[25]: # Step 1: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
```

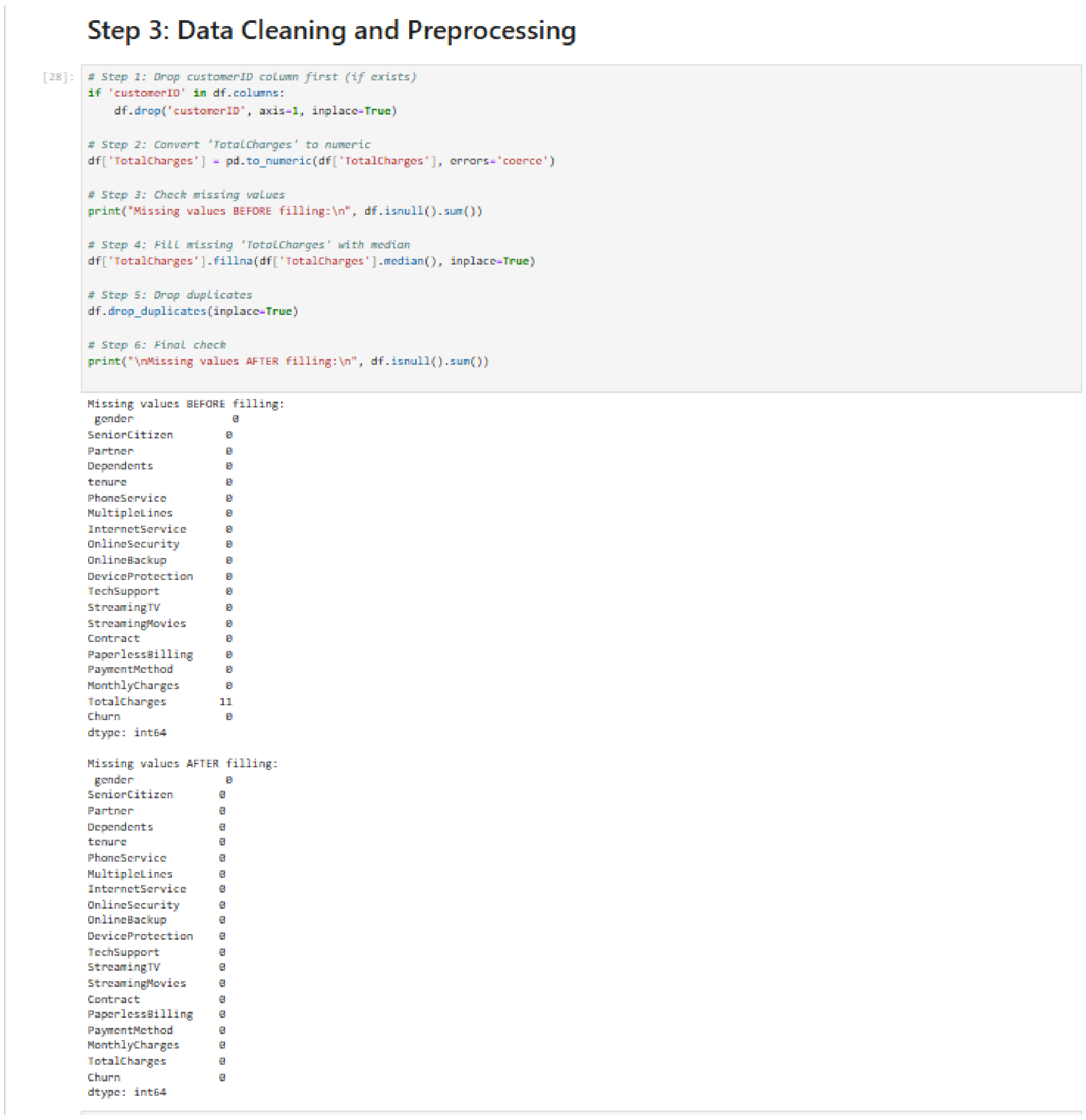


Step 2: Load Dataset

```
]: import pandas as pd

df = pd.read_csv("telco_churn_data/WA_Fn-UseC_-Telco-Customer-Churn.csv")
df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	St
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	

5 rows × 21 columns

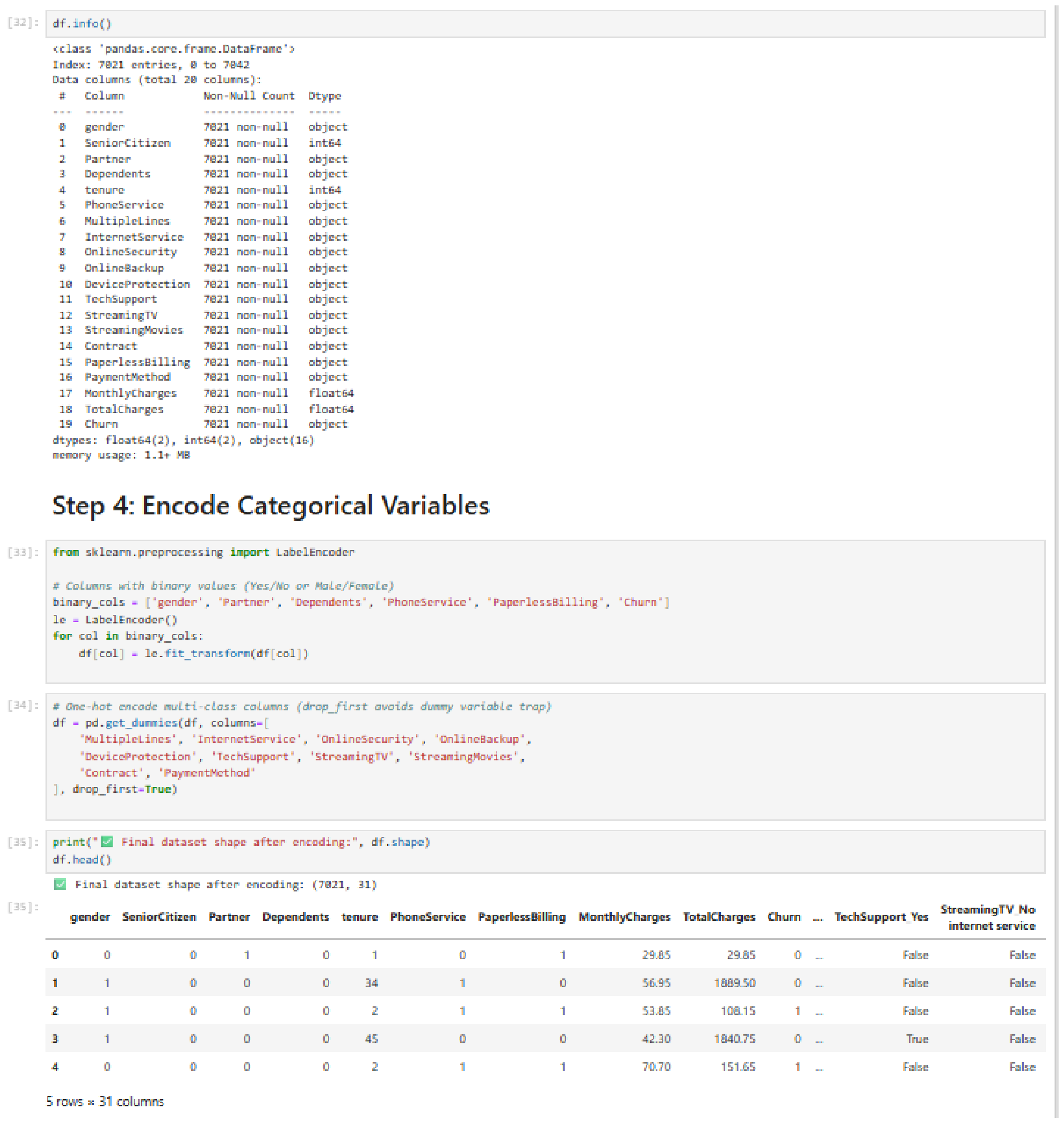


Step 3: Data Cleaning and Preprocessing

[28]:

```
# Step 1: Drop customerID column first (if exists)
if 'customerID' in df.columns:
    df.drop('customerID', axis=1, inplace=True)

# Step 2: Convert 'TotalCharges' to numeric
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Step 3: Check missing values
print("Missing values BEFORE filling:\n", df.isnull().sum())

# Step 4: Fill missing 'TotalCharges' with median
df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)

# Step 5: Drop duplicates
df.drop_duplicates(inplace=True)

# Step 6: Final check
print("\nMissing values AFTER filling:\n", df.isnull().sum())
```

```
Missing values BEFORE filling:
 gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64

Missing values AFTER filling:
 gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```



```
[32]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7021 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7021 non-null   object
 1   SeniorCitizen     7021 non-null   int64
 2   Partner           7021 non-null   object
 3   Dependents        7021 non-null   object
 4   tenure            7021 non-null   int64
 5   PhoneService      7021 non-null   object
 6   MultipleLines     7021 non-null   object
 7   InternetService   7021 non-null   object
 8   OnlineSecurity    7021 non-null   object
 9   OnlineBackup      7021 non-null   object
 10  DeviceProtection  7021 non-null   object
 11  TechSupport       7021 non-null   object
 12  StreamingTV       7021 non-null   object
 13  StreamingMovies   7021 non-null   object
 14  Contract          7021 non-null   object
 15  PaperlessBilling  7021 non-null   object
 16  PaymentMethod     7021 non-null   object
 17  MonthlyCharges    7021 non-null   float64
 18  TotalCharges      7021 non-null   float64
 19  Churn             7021 non-null   object
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

Step 4: Encode Categorical Variables

```
[33]: from sklearn.preprocessing import LabelEncoder

      # Columns with binary values (Yes/No or Male/Female)
      binary_cols = ['gender', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Churn']
      le = LabelEncoder()
      for col in binary_cols:
          df[col] = le.fit_transform(df[col])
```

```
[34]: # One-hot encode multi-class columns (drop_first avoids dummy variable trap)
      df = pd.get_dummies(df, columns=[
          'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
          'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
          'Contract', 'PaymentMethod'
      ], drop_first=True)
```

```
[35]: print("✅ Final dataset shape after encoding:", df.shape)
      df.head()
```

```
✅ Final dataset shape after encoding: (7021, 31)
```

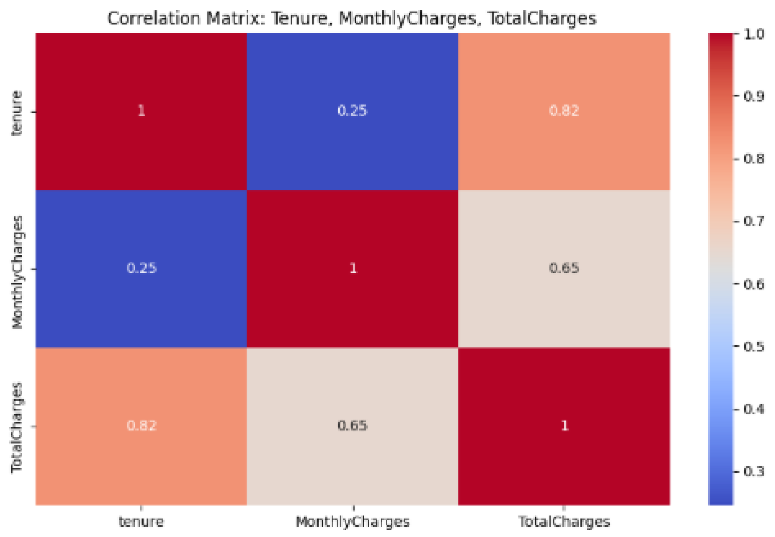
[35]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	PaperlessBilling	MonthlyCharges	TotalCharges	Churn	...	TechSupport_Yes	StreamingTV_No internet service
0	0	0	1	0	1	0	1	29.85	29.85	0	...	False	False
1	1	0	0	0	34	1	0	56.95	1889.50	0	...	False	False
2	1	0	0	0	2	1	1	53.85	108.15	1	...	False	False
3	1	0	0	0	45	0	0	42.30	1840.75	0	...	True	False
4	0	0	0	0	2	1	1	70.70	151.65	1	...	False	False

5 rows × 31 columns

Heatmap: Correlation Between Numerical Fields

[37]:
```python
# Correlation Matrix (Numerical Columns Only)
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.heatmap(df[['tenure', 'MonthlyCharges', 'TotalCharges']].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix: Tenure, MonthlyCharges, TotalCharges")
plt.show()
```

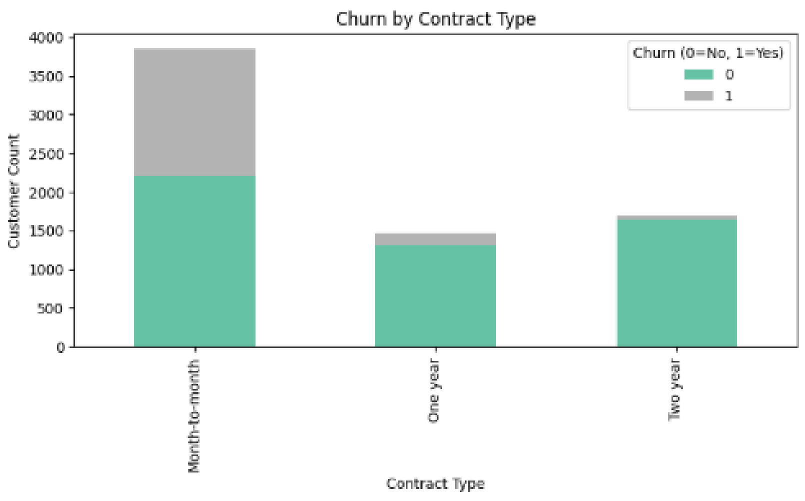


Bar Chart: Churn (0/1) vs. Contract Type

[49]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Derive Contract_Month-to-month column
df['Contract_Month-to-month'] = 1 - df['Contract_One year'] - df['Contract_Two year']

# Convert contract columns to int for idxmax
contract_cols = ['Contract_Month-to-month', 'Contract_One year', 'Contract_Two year']
df['Contract_Type'] = df[contract_cols].astype(int).idxmax(axis=1)

# Map to readable labels
contract_map = {
    'Contract_Month-to-month': 'Month-to-month',
    'Contract_One year': 'One year',
    'Contract_Two year': 'Two year'
}
df['Contract_Type'] = df['Contract_Type'].map(contract_map)

# Plot stacked bar chart
contract_churn = df.groupby(['Contract_Type', 'Churn']).size().unstack()
contract_churn.plot(kind='bar', stacked=True, figsize=(8, 5), colormap='Set2')
plt.title("Churn by Contract Type")
plt.xlabel("Contract Type")
plt.ylabel("Customer Count")
plt.legend(title="Churn (0=No, 1=Yes)")
plt.tight_layout()
plt.show()
```

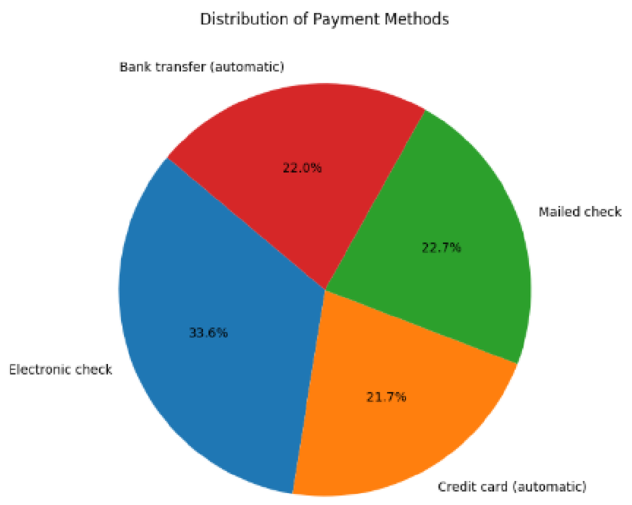
Bar Chart: Churn (0/1) vs. Contract Type

[49]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Derive Contract_Month-to-month column
df['Contract_Month-to-month'] = 1 - df['Contract_One year'] - df['Contract_Two year']

# Convert contract columns to int for idxmax
contract_cols = ['Contract_Month-to-month', 'Contract_One year', 'Contract_Two year']
df['Contract_Type'] = df[contract_cols].astype(int).idxmax(axis=1)

# Map to readable labels
contract_map = {
    'Contract_Month-to-month': 'Month-to-month',
    'Contract_One year': 'One year',
    'Contract_Two year': 'Two year'
}
df['Contract_Type'] = df['Contract_Type'].map(contract_map)

# Plot stacked bar chart
contract_churn = df.groupby(['Contract_Type', 'Churn']).size().unstack()
contract_churn.plot(kind='bar', stacked=True, figsize=(8, 5), colormap='Set2')
plt.title("Churn by Contract Type")
plt.xlabel("Contract Type")
plt.ylabel("Customer Count")
plt.legend(title="Churn (0=No, 1=Yes)")
plt.tight_layout()
plt.show()
```



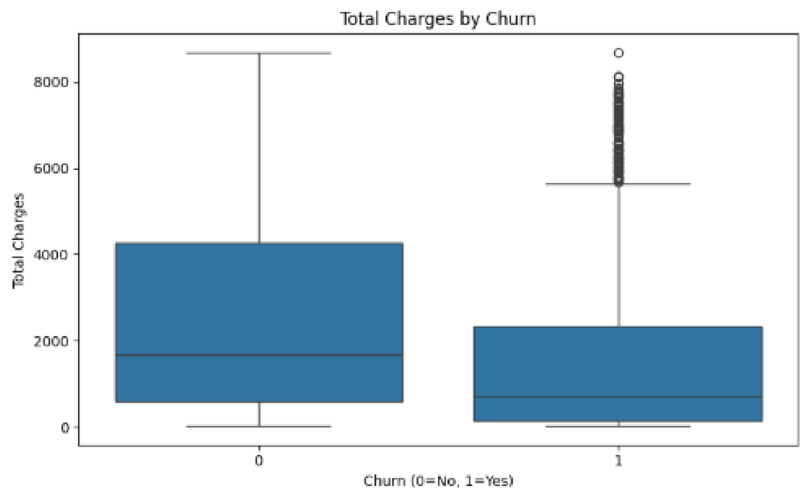
Pie Chart: Payment Method

```
[43]: # Calculate original counts
pm_counts = {
    'Electronic check': df['PaymentMethod_Electronic check'].sum(),
    'Credit card (automatic)': df['PaymentMethod_Credit card (automatic)'].sum(),
    'Mailed check': df['PaymentMethod_Mailed check'].sum(),
    'Bank transfer (automatic)': len(df) - (
        df['PaymentMethod_Electronic check'] +
        df['PaymentMethod_Credit card (automatic)'] +
        df['PaymentMethod_Mailed check']
    ).sum()
}

# Pie chart
plt.figure(figsize=(7, 7))
plt.pie(pm_counts.values(), labels=pm_counts.keys(), autopct='%1.1f%%', startangle=140)
plt.title("Distribution of Payment Methods")
plt.tight_layout()
plt.show()
```



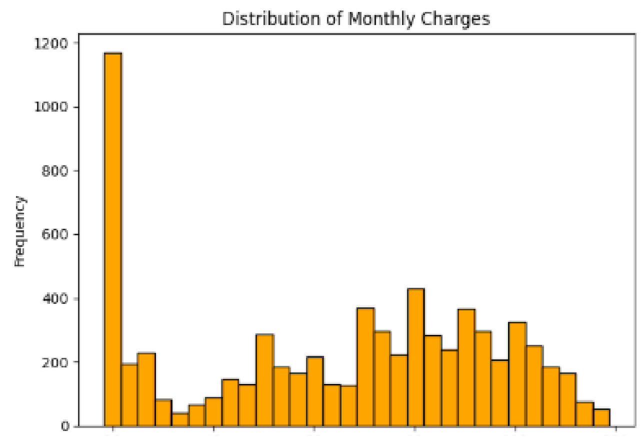
Box Plot: TotalCharges by Churn (0/1)

```
[67]: import seaborn as sns

plt.figure(figsize=(8, 5))
sns.boxplot(x='Churn', y='TotalCharges', data=df)
plt.title("Total Charges by Churn")
plt.xlabel("Churn (0=No, 1=Yes)")
plt.ylabel("Total Charges")
plt.tight_layout()
plt.show()
```



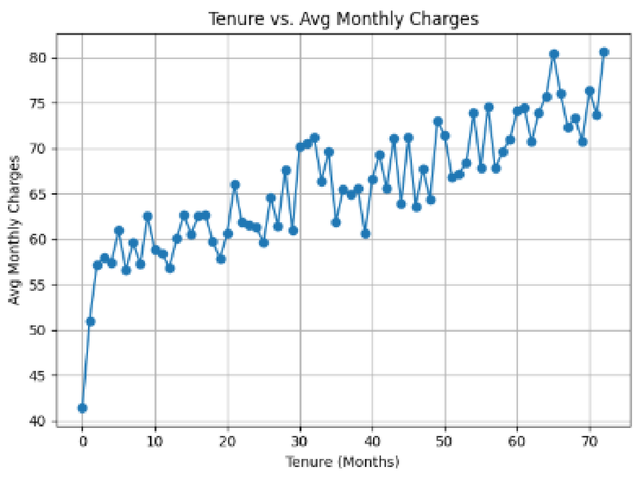
Histogram: Monthly Charges

```
[45]: plt.hist(df['MonthlyCharges'], bins=30, edgecolor='black', color='orange')
plt.title("Distribution of Monthly Charges")
plt.xlabel("Monthly Charges")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```



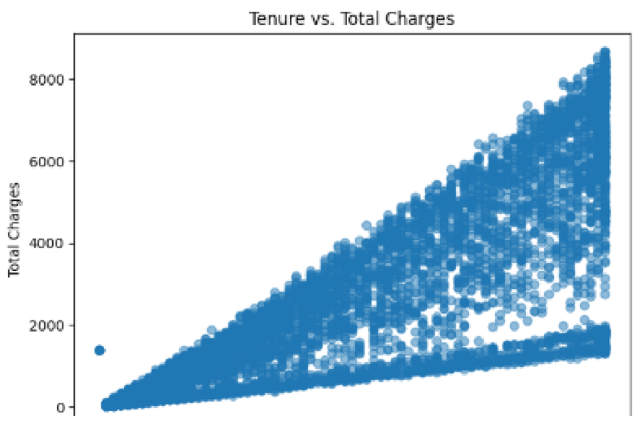
Line Graph: Tenure vs. Avg Monthly Charges

[46]:
```
df_line = df.groupby('tenure')['MonthlyCharges'].mean().reset_index()

plt.plot(df_line['tenure'], df_line['MonthlyCharges'], marker='o')
plt.title("Tenure vs. Avg Monthly Charges")
plt.xlabel("Tenure (Months)")
plt.ylabel("Avg Monthly Charges")
plt.grid(True)
plt.tight_layout()
plt.show()
```



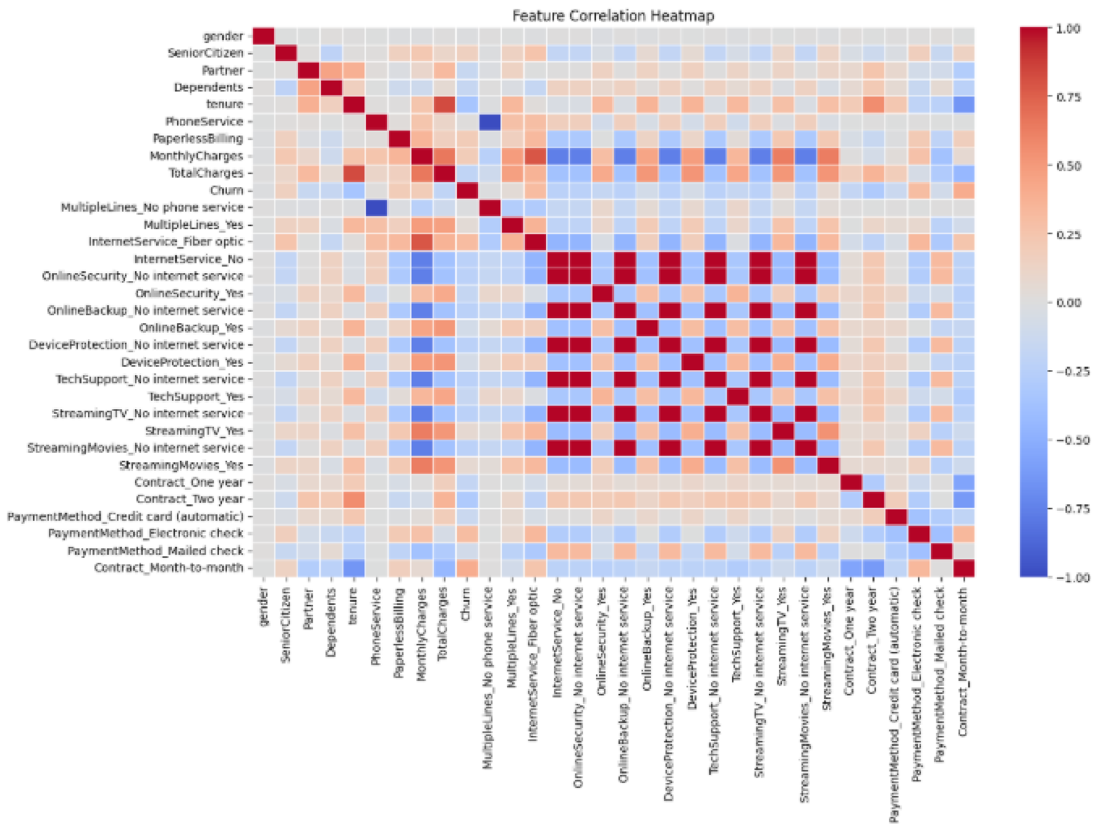
Scatter Plot: Tenure vs. TotalCharges

[47]:
```
plt.scatter(df['tenure'], df['TotalCharges'], alpha=0.5)
plt.title("Tenure vs. Total Charges")
plt.xlabel("Tenure (Months)")
plt.ylabel("Total Charges")
plt.tight_layout()
plt.show()
```



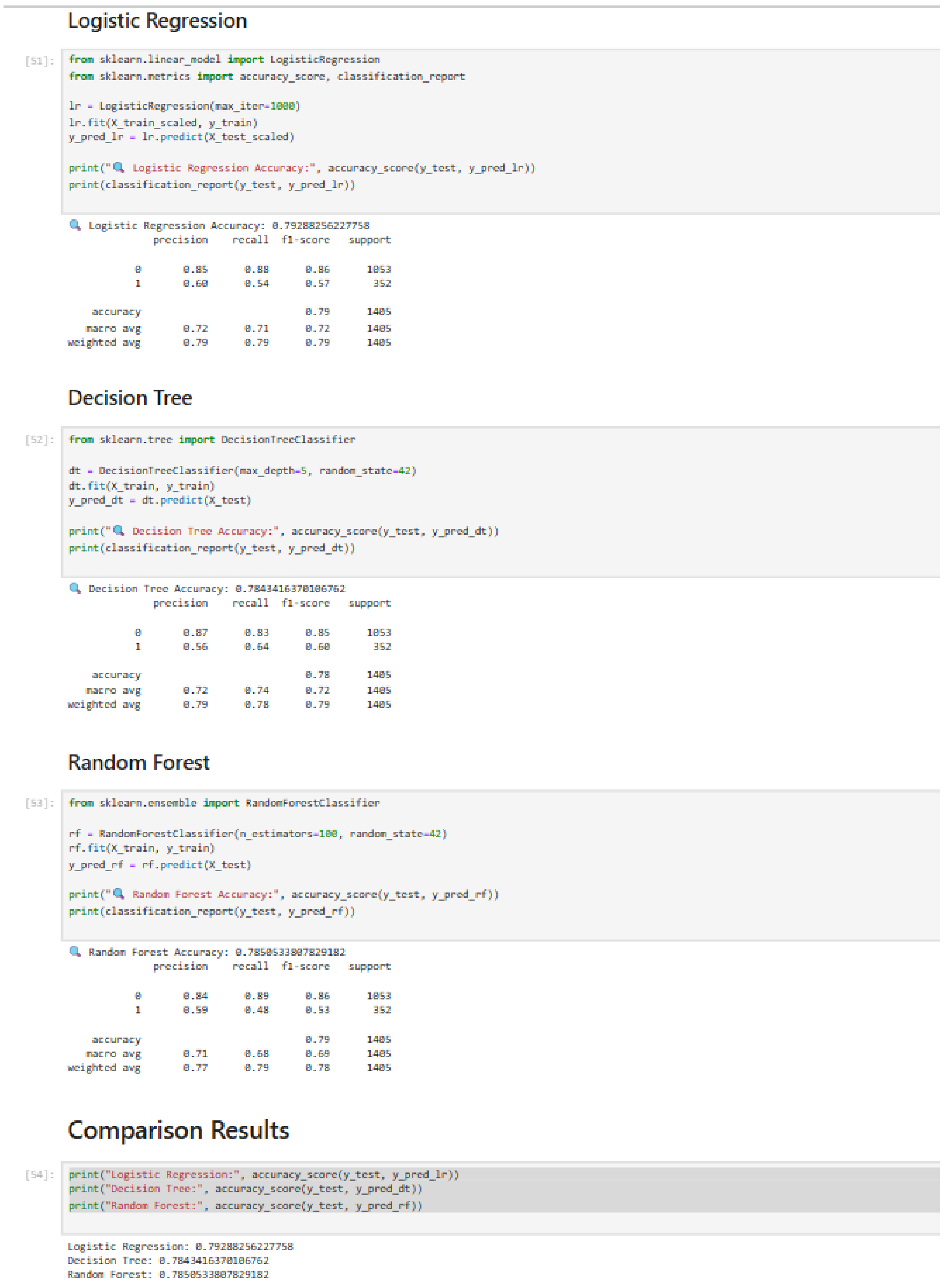
Heatmap of All Correlations

[48]:
```
plt.figure(figsize=(14, 10))
sns.heatmap(df.corr(), cmap='coolwarm', linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.tight_layout()
plt.show()
```



Data Mining and Machine Learning

[50]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Features and target
X = df.drop(['Churn', 'Contract_Type'], axis=1)
y = df['Churn']

# Train-test split (80-20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features (except for tree-based models)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```



Logistic Regression

```
[51]: from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import accuracy_score, classification_report

      lr = LogisticRegression(max_iter=1000)
      lr.fit(X_train_scaled, y_train)
      y_pred_lr = lr.predict(X_test_scaled)

      print("🔍 Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
      print(classification_report(y_test, y_pred_lr))
```

```
🔍 Logistic Regression Accuracy: 0.79288256227758
              precision    recall  f1-score   support

           0       0.85      0.88      0.86      1053
           1       0.60      0.54      0.57       352

    accuracy                           0.79      1405
   macro avg       0.72      0.71      0.72      1405
weighted avg       0.79      0.79      0.79      1405
```

Decision Tree

```
[52]: from sklearn.tree import DecisionTreeClassifier

      dt = DecisionTreeClassifier(max_depth=5, random_state=42)
      dt.fit(X_train, y_train)
      y_pred_dt = dt.predict(X_test)

      print("🔍 Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
      print(classification_report(y_test, y_pred_dt))
```

```
🔍 Decision Tree Accuracy: 0.7843416370106762
              precision    recall  f1-score   support

           0       0.87      0.83      0.85      1053
           1       0.56      0.64      0.60       352

    accuracy                           0.78      1405
   macro avg       0.72      0.74      0.72      1405
weighted avg       0.79      0.78      0.79      1405
```

Random Forest

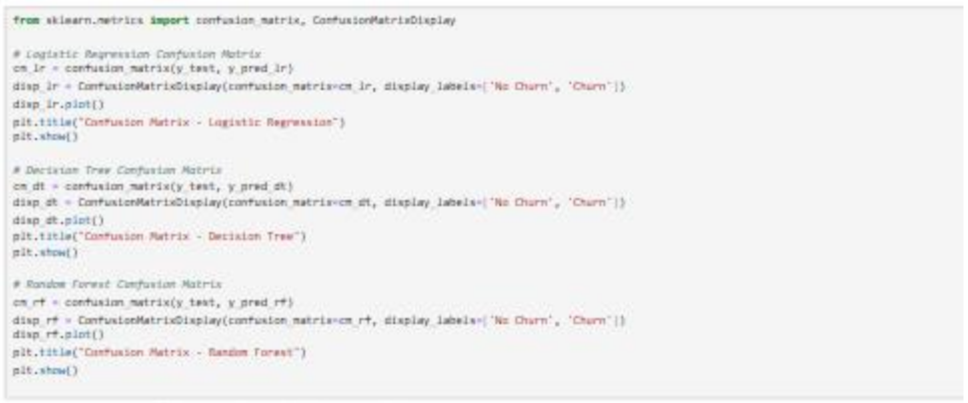
```
[53]: from sklearn.ensemble import RandomForestClassifier

      rf = RandomForestClassifier(n_estimators=100, random_state=42)
      rf.fit(X_train, y_train)
      y_pred_rf = rf.predict(X_test)

      print("🔍 Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
      print(classification_report(y_test, y_pred_rf))
```

```
🔍 Random Forest Accuracy: 0.7850533807829182
              precision    recall  f1-score   support

           0       0.84      0.89      0.86      1053
           1       0.59      0.48      0.53       352

    accuracy                           0.79      1405
   macro avg       0.71      0.68      0.69      1405
weighted avg       0.77      0.79      0.78      1405
```

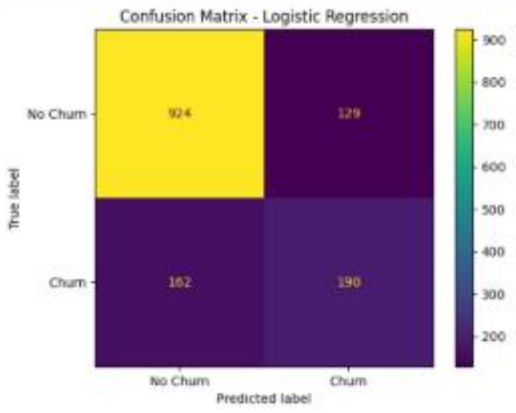
Comparison Results

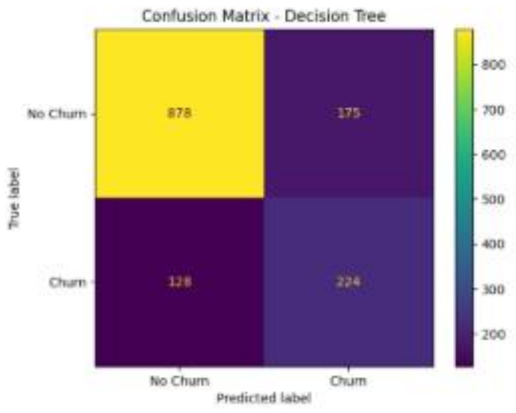
```
[54]: print("Logistic Regression:", accuracy_score(y_test, y_pred_lr))
      print("Decision Tree:", accuracy_score(y_test, y_pred_dt))
      print("Random Forest:", accuracy_score(y_test, y_pred_rf))
```

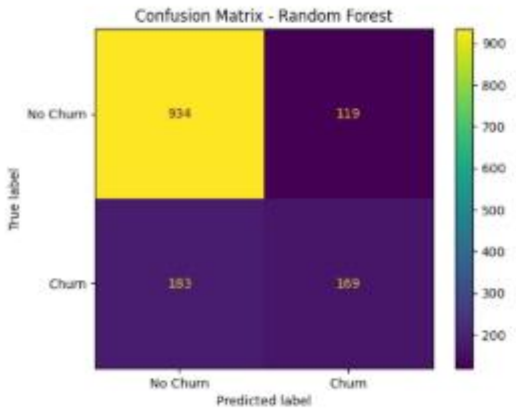
```
Logistic Regression: 0.79288256227758
Decision Tree: 0.7843416370106762
Random Forest: 0.7850533807829182
```



```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Logistic Regression Confusion Matrix
cm_lr = confusion_matrix(y_test, y_pred_lr)
disp_lr = ConfusionMatrixDisplay(confusion_matrix=cm_lr, display_labels=['No Churn', 'Churn'])
disp_lr.plot()
plt.title("Confusion Matrix - Logistic Regression")
plt.show()

# Decision Tree Confusion Matrix
cm_dt = confusion_matrix(y_test, y_pred_dt)
disp_dt = ConfusionMatrixDisplay(confusion_matrix=cm_dt, display_labels=['No Churn', 'Churn'])
disp_dt.plot()
plt.title("Confusion Matrix - Decision Tree")
plt.show()

# Random Forest Confusion Matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)
disp_rf = ConfusionMatrixDisplay(confusion_matrix=cm_rf, display_labels=['No Churn', 'Churn'])
disp_rf.plot()
plt.title("Confusion Matrix - Random Forest")
plt.show()
```

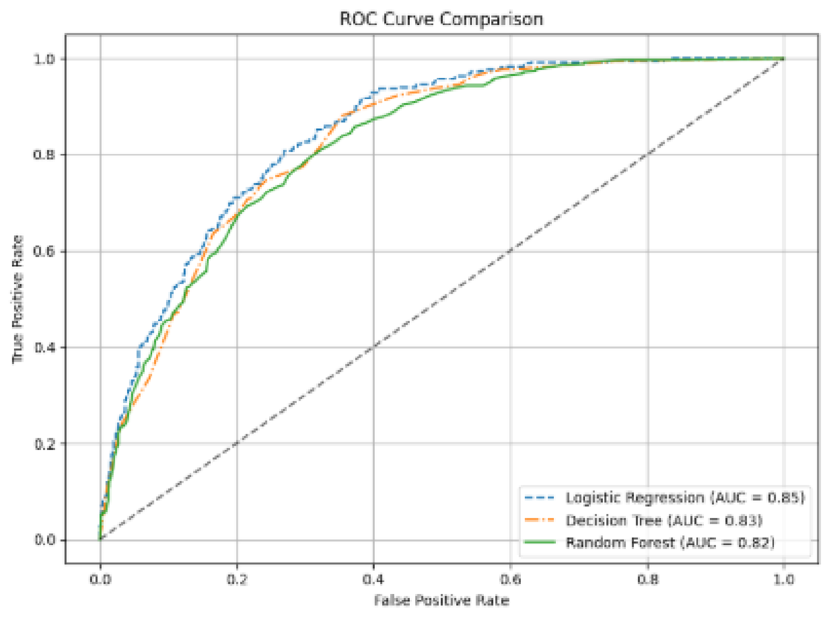






[56]:

```python
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Logistic Regression
y_prob_lr = lr.predict_proba(X_test_scaled)[:, 1]
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_prob_lr)
auc_lr = roc_auc_score(y_test, y_prob_lr)

# Decision Tree
y_prob_dt = dt.predict_proba(X_test)[:, 1]
fpr_dt, tpr_dt, _ = roc_curve(y_test, y_prob_dt)
auc_dt = roc_auc_score(y_test, y_prob_dt)

# Random Forest
y_prob_rf = rf.predict_proba(X_test)[:, 1]
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_prob_rf)
auc_rf = roc_auc_score(y_test, y_prob_rf)

# Plot all on one graph
plt.figure(figsize=(8, 6))
plt.plot(fpr_lr, tpr_lr, label=f"Logistic Regression (AUC = {auc_lr:.2f})", linestyle='--')
plt.plot(fpr_dt, tpr_dt, label=f"Decision Tree (AUC = {auc_dt:.2f})", linestyle='-.')
plt.plot(fpr_rf, tpr_rf, label=f"Random Forest (AUC = {auc_rf:.2f})", linestyle='-')

plt.plot([0, 1], [0, 1], 'k--', alpha=0.6)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve Comparison")
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
```



[58]:

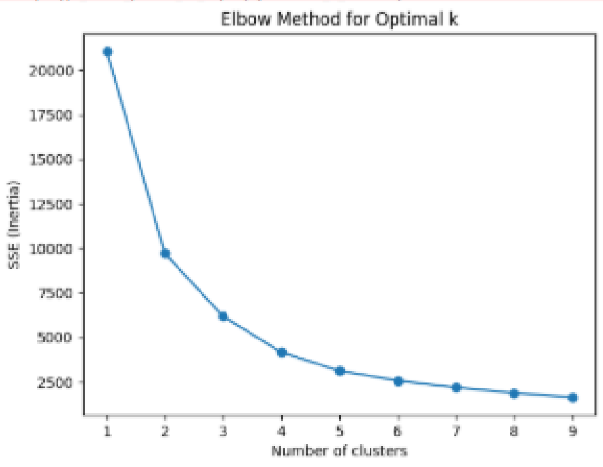
```python
from sklearn.metrics import roc_auc_score

# AUC values
auc_lr = roc_auc_score(y_test, y_prob_lr)
auc_dt = roc_auc_score(y_test, y_prob_dt)
auc_rf = roc_auc_score(y_test, y_prob_rf)

print("Logistic Regression AUC:", round(auc_lr, 3))
print("Decision Tree AUC:", round(auc_dt, 3))
print("Random Forest AUC:", round(auc_rf, 3))
```

```
Logistic Regression AUC: 0.846
Decision Tree AUC: 0.83
Random Forest AUC: 0.822
```

[59]:

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt

# Select relevant features
X_clust = df[['tenure', 'MonthlyCharges', 'TotalCharges']]

# Scale data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_clust)

# Elbow method to find optimal clusters
sse = []
for k in range(1, 10):
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(X_scaled)
    sse.append(km.inertia_)

plt.plot(range(1, 10), sse, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('SSE (Inertia)')
plt.title('Elbow Method for Optimal k')
plt.show()
```

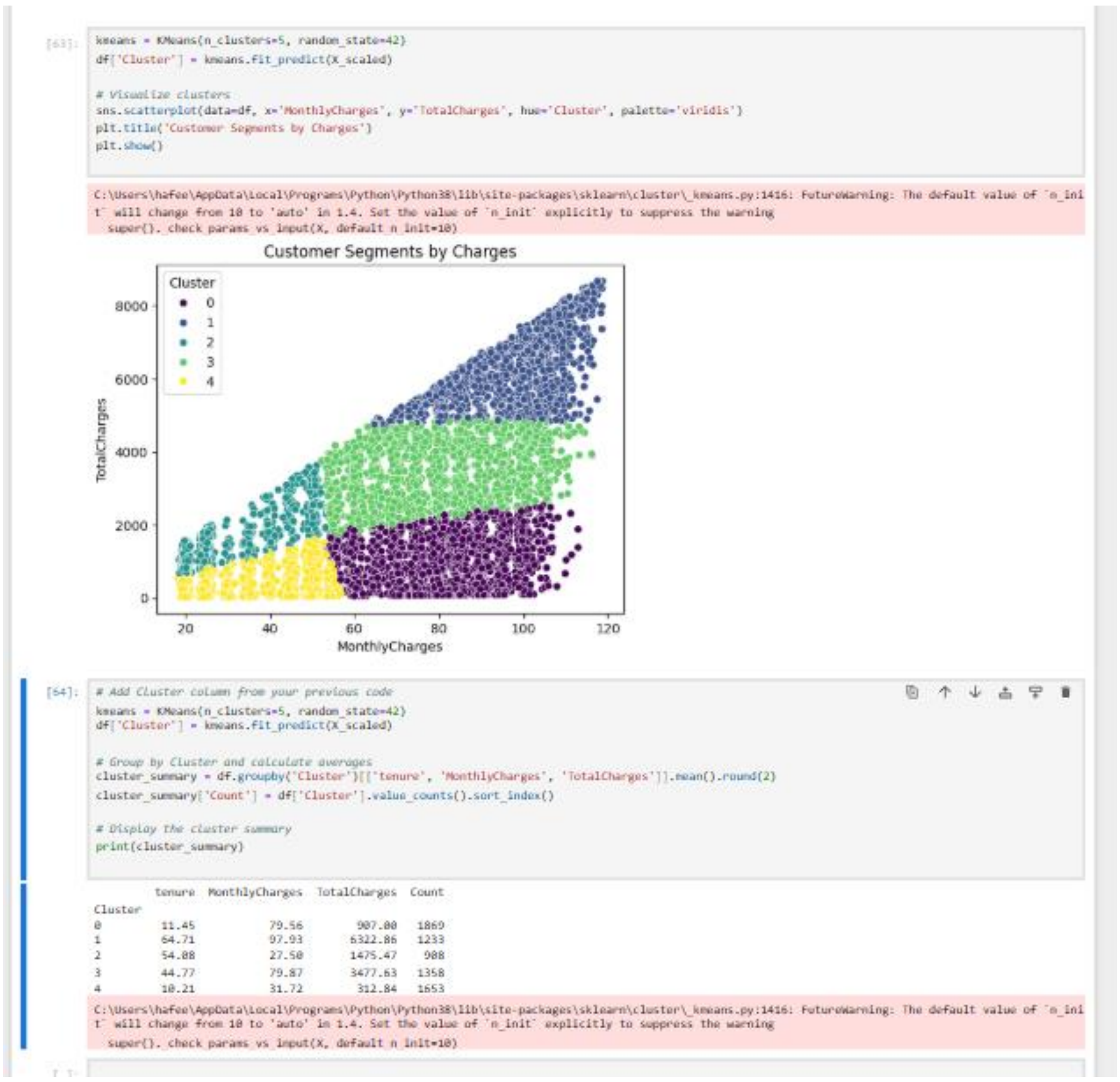
```
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



[63]:

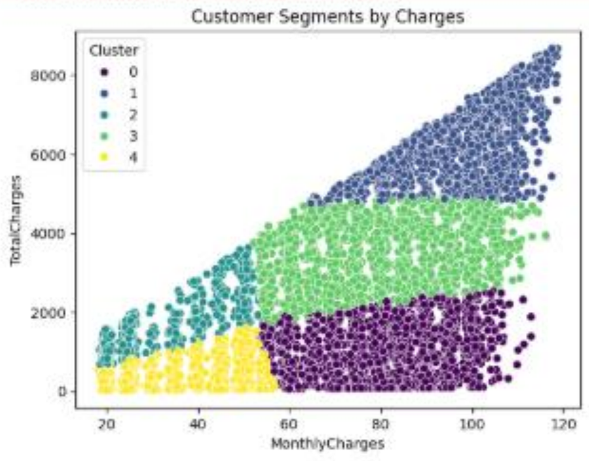
```
kmeans = KMeans(n_clusters=5, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)

# Visualize clusters
sns.scatterplot(data=df, x='MonthlyCharges', y='TotalCharges', hue='Cluster', palette='viridis')
plt.title('Customer Segments by Charges')
plt.show()
```

```
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



```
[63]: kmeans = KMeans(n_clusters=5, random_state=42)
      df['Cluster'] = kmeans.fit_predict(X_scaled)

      # Visualize clusters
      sns.scatterplot(data=df, x='MonthlyCharges', y='TotalCharges', hue='Cluster', palette='viridis')
      plt.title('Customer Segments by Charges')
      plt.show()
```

```
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



```
[64]: # Add Cluster column from your previous code
      kmeans = KMeans(n_clusters=5, random_state=42)
      df['Cluster'] = kmeans.fit_predict(X_scaled)

      # Group by Cluster and calculate averages
      cluster_summary = df.groupby('Cluster')[['tenure', 'MonthlyCharges', 'TotalCharges']].mean().round(2)
      cluster_summary['Count'] = df['Cluster'].value_counts().sort_index()

      # Display the cluster summary
      print(cluster_summary)
```

```
         tenure  MonthlyCharges  TotalCharges  Count
Cluster
0         11.45           79.56        907.00   1869
1         64.71           97.93       6322.86   1233
2         54.88           27.50       1475.47    988
3         44.77           79.87       3477.63   1358
4         10.21           31.72        312.84   1653
C:\Users\hafee\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
[ ]:
```