ROBERT GORDON UNIVERSITY ABERDEEN

INFORMATICS INSTITUTE OF TECHNOLOGY

BSc (Hons) Artificial Intelligence and Data Science

Individual Coursework Report

Module: CM2604 Machine Learning

Module Leader: Dr. Sahan Priyanayana

RGU Student ID: 2507578

IIT Student ID: 20240046

Student Name: S. Gowrisankar

Acknowledgement

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this report. My sincere appreciation goes to our lecturers Dr. Sahan Priyanayana, Ms. Rajeiha Sutharshan and Ms. Sadia Hussain for their invaluable assistance and guidance throughout the semester and preparation of this report.

I would also like to express our deepest gratitude to IIT for facilitating us with the internet, library, and classroom facilities.

Last but not least, I would like to express our gratitude to all our family members for all their love and patience in making this journey a success.

Executive Summary

This report presents a complete Machine Learning (ML) pipeline for predicting customer churn using the Telco Customer Churn dataset. The workflow include Exploratory Data Analysis (EDA), feature engineering, data preprocessing and implementation of two supervised machine learning models, a Neural Network and Decision Tree classifier. Ethical consideration including fairness, transparency and responsible data handling were integrated throughout the development lifecycle. Post deployment strategies are also proposed to ensure the model reliability, compliance and fairness over time. The report concludes with limitations and recommendations for future improvements.

Table of Contents

List of Figures

Telco Customer Churn Prediction Using Machine Learning

Introduction

This project focuses on developing a complete machine learning solution to predict customer churn in the telecommunications sector. Customer churn refers to when a customer discontinues their service. Being able to accurately predict churn allows the company to identify high risk customers and take proactive retention measures.

Using the Telco Customer Churn dataset, this project build an pipeline that:

- Analyses customer behavior through Exploratory Data Analysis (EDA)
- Preprocesses and transforms the raw customer data into ML ready features
- Implements two supervised learning models (Neural Network and Decision Tree)
- Evaluates and compares the models
- Considers the ethical, legal and social factors related to data usage and model deployment

The outcome of this project is a functional churn prediction framework that can support telecom providers in improving customer retention strategies and understanding the underlying factors contributing to the churn.

Corpus Preparation and Dataset Overview

The dataset used in this project is publicly available in Kaggle.

1. Dataset Information

- Number of rows – 7043
- Number of columns – 21
- Target variable – Churn (Yes/No)
- Feature Types:
    - Categorical (Contract, PaymentMethod, etc.)
    - Binary (Yes/No)
    - Numerical (MonthlyCharges, TotalCharges, etc.)

2. Dataset Upload and Initial Structure

The dataset was uploaded through the file upload widget in Google Colab. The initial structure was examined by using df.head(), df.info(), missing value check and the the ctagorical summary.

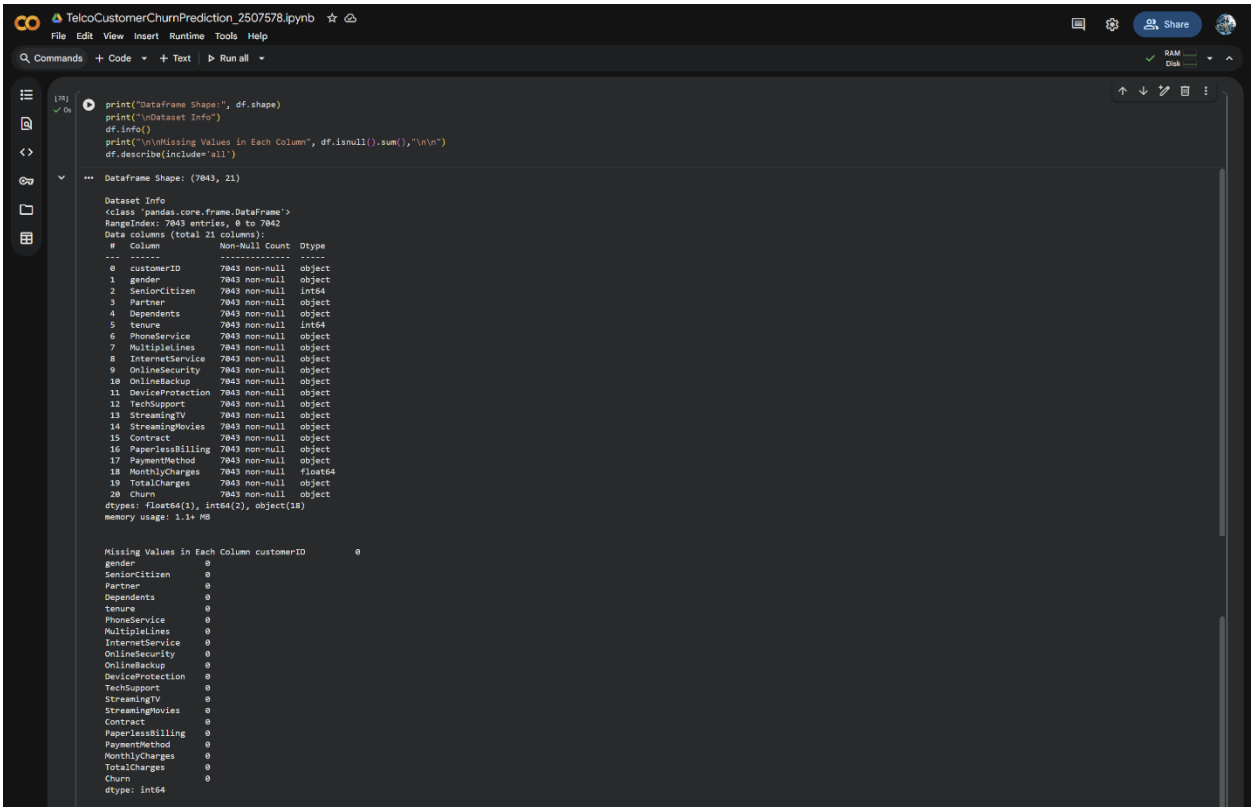

Figure 1 - Dataset frame size, columns and missing values info

Figure 2 - Categorical Summary

Task 1 – Exploratory Data Analysis (EDA)

The EDA provides an insight into the dataset's structure, feature distributions and correlations.

1. Handling Missing Data (Data Cleaning)

The column "TotalCharges" contained blank strings. These were safely converted using pd.to_numeric(errors='coerce'). Rows with missing values in "TotalCharges" were removed.

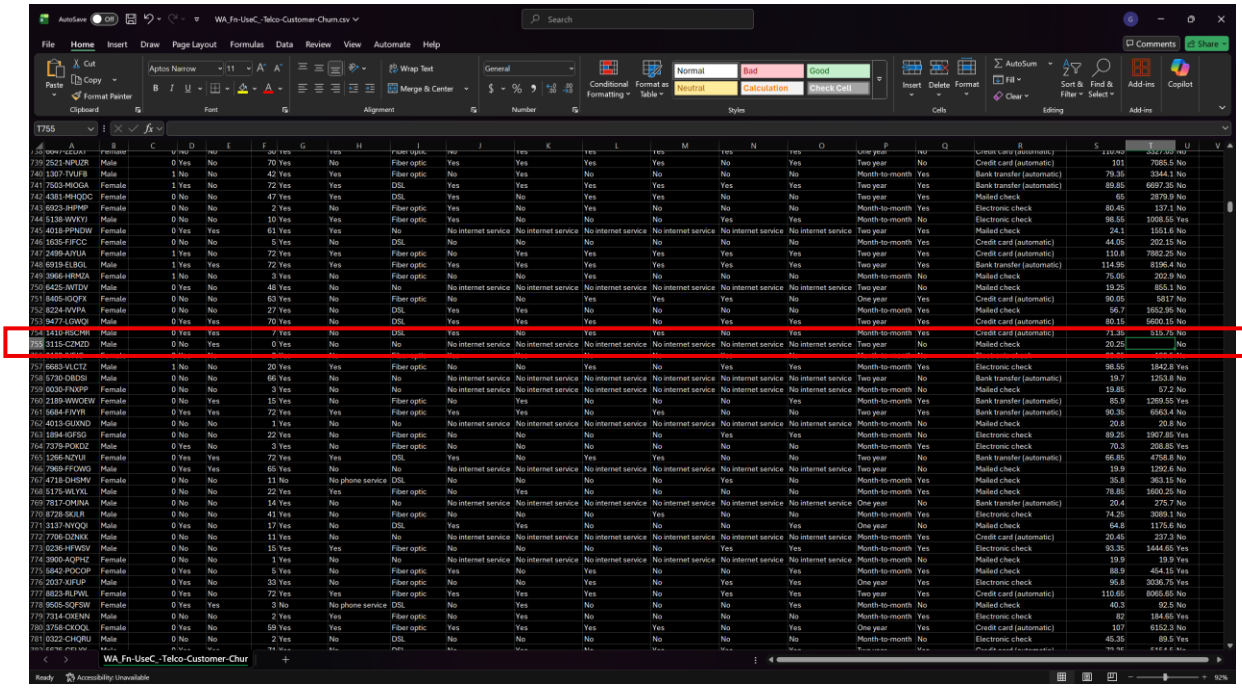

Figure 3 - Blank Cell in Dataset

```
#convert TotalCharges to numeric as some fields contain empty strings
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

#drop rows where TotalCharges is NaN (not a huge probelm as only a very few are there)
df = df.dropna(subset=['TotalCharges'])

#reset index after deletion
df.reset_index(drop=True, inplace=True)

print("Dataset shape after cleaning:", df.shape)
```

```
Dataset shape after cleaning: (7032, 21)
```

Figure 4 - Data Cleaning Process

2. EDA Visualizations

2.1. Churn Distribution

The dataset seems imbalanced with more non churning option being prominent compared to the churn rate.

2.2. Monthly Charges Distribution

The histogram of Monthly Charges reveals that a large proportion of customers fall within the low-cost tier (USD 18 -25), forming the highest peak. Additional peaks between USD 70-90 reflect mid-tier subscription packages. This indicates that the telecom service provider user tiered pricing models which results in distinct customer segments. Understanding these tiers is very important as customers in higher pricing brackets frequently exhibit more volatility in churn behaviors.

The key insight here is that the monthly charges vary widely and customers paying higher amounts may be more likely to churn, aligning with established patterns in telecom churn behaviour.

2.3. Contract Type Vs. Churn

The visualization comparing the churn across contract types shows a dramatic difference in churn behavior. Month to month customers has the highest churn rate, likely due to no long term commitment. One year contract customers show a significantly lower churn rate while the two year contract customers show the lowest churn rate of all. This confirms that the contract type is one of the strongest predictors of churn rate and customers under flexible agreements tend to leave more easily compared the long term ones.

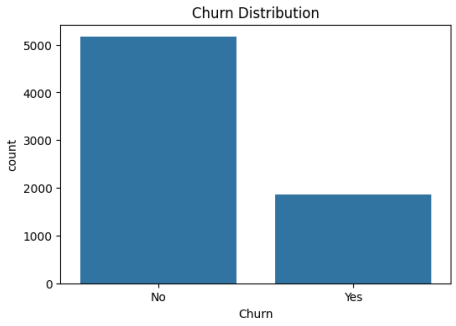

Figure 5 - Churn Distribution

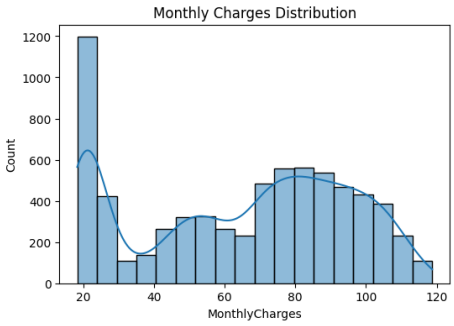

Figure 6 - Monthly Charges Distribution

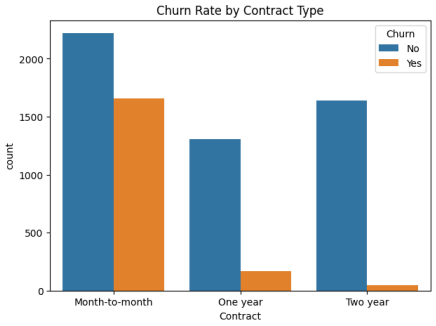

Figure 7 - Contract Type Vs. Churn

2.4. Correlation Heatmap

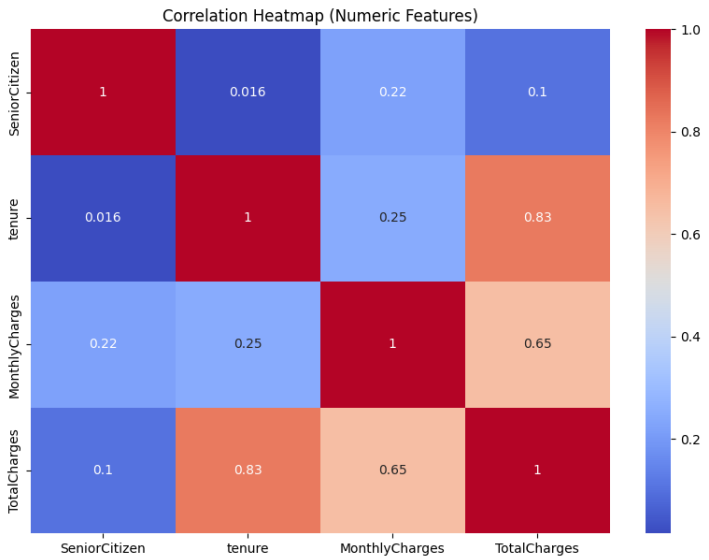

Figure 8 - Correlation Heatmap

The correlation heatmap shows several important relationships. The 'Tenure' and 'TotalCharges' show a strong positive correlation of 0.83 as long term customers logically accumulate higher billing. The 'MonthlyCharges' and 'TotalCharges' shows a moderate correlation of 0.65 indicating monthly fees substantially contribute to total spending. Meanwhile, 'SeniorCitizen' exhibits the weakest correlation with other features indicating that age alone is not a dominant factor in churn.

3. Insights Extracted

- The dataset is imbalanced (more customers did not churn)
- The Month-to-Month contracts indicate unstable customer retention
- High monthly charges contribute to churn risk
- The customer's age does not meaningfully drive outcomes
- No severe missing value issues

Solution Methodology

1. Algorithm Selection

Two supervised Machine Learning (ML) models were use in this project.

1. Neural Network Model (TensorFlow)
2. Decision Tree Classifier Model

2. Optimizer Choice (Adam)

The Adam optimizer was selected due to its strong performance in training Neural Network on tubular datasets. The reasons for choosing Adam includes:

1. Automatically adjusts learning rates which reduces manual tuning
2. Computationally efficient and converges faster than standard Stochastic Gradient Descend
3. Adam is widely adopted for binary classification and is considered a strong baseline in most ML workflows

3. Loss Function (Binary Crossentropy)

Binary Crossentropy was chosen because it is the standard loss function for binary classification tasks. It measures the difference between predicted probabilities and actual class labels. Moreover, it penalizes confident incorrect prediction more strongly which helps in better separation between churn and non churn classes. Thus, Binary Crossentropy is mathematically appropriate for modelling churn prediction probabilities.

4. Preprocessing Strategy

4.1. Binary Encoding

Binary fields like yes/no were mapped to 1/0 as it creates clean numerical inputs for the ML algorithms and avoids unnecessary expansion of dimensions through one hot encoding

4.2. One Hot Encoding

One Hot Encoding was chosen for multi category features as many algorithms cannot interpret text categories directly. This method preserves all category information without introducing any arbitrary numeric relationships. Moreover, this also avoids bias from ordinal encodings.

4.3. Feature Scaling

Feature Scaling with 'StandardScaler' was applied to numerical features as Neural Networks require scaled input for stable gradient descent. 'StandardScaler' prevents features with large ranges dominating over smaller scale features. This results in an improved convergence speed.

4.4. Evaluation Criteria

Models are evaluated using their accuracy, precision, recall, F1 score and confusion matrix. These metrics provide insight into both correctness and class level performance

4.5. Hyperparameter Tuning

A 'GridSearchCV' tuning approach was used for the Decision Tree Classifier Model beacause it exhaustively tests combinations of parameters to find the best performing model as Decision Trees are sensitive to parameters such as max depth, criterion and minimum sample split. Also, cross validation ensures evaluation fairness and reduces overfitting.

The choice of the tuned parameters are explained below:

- max_depth – Controls model complexity and prevents overfitting
- criterion (gini and entropy) – These tests both impurity metrics to determine which splits best for the churn
- min_sample_split – Prevents overly specific splits and improves generalization

Task 2 – Data Preprocessing and Model Development

1. Data Preprocessing

1.1. Encoding

Binary Yes/No columns mapped to 1/0 and Multi category columns were one hot encoded

1.2. Feature Scaling

"StandardScaler" was applied to all numerical features. This is to improve the Neural Network performance.

1.3. Train/Test Split

An 80:20 split was done and stratified sampling is used to maintain churn class ratios

```
from sklearn.preprocessing import LabelEncoder, StandardScaler

df_processed = df.copy()

#removing customerID for this process as its a unique identofier and not a feature
df_processed=df_processed.drop(['customerID'], axis=1)

#encode yes/no columns and making them binary values
binary_cols = ['Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Churn']
for col in binary_cols:
    df_processed[col] = df_processed[col].map({'Yes': 1, 'No': 0})

#one hot encode multi category categorical columns
df_processed = pd.get_dummies(df_processed, drop_first=True)

#separate features and labels
X = df_processed.drop('Churn', axis=1)
y = df_processed['Churn']

#scale numerical data for neural network model
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Data Preprocessing completed!")
print("Number of features:", X_scaled.shape[1])
```

```
Data Preprocessing completed!
Number of features: 30
```

Figure 9 - Encoding and Feature Scaling

```
from sklearn.model_selection import train_test_split

#spltiing data to 80:20 ratio for training and testing
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y)

print("Training samples:", len(X_train))
print("Testing samples:", len(X_test))
```

```
Training samples: 5625
Testing samples: 1407
```

Figure 10 - Train/Test Split

2. Model 1 – Neural Network Classifier Using TensorFlow

2.1. Model Architecture

- Input Layer – Scaled features
- Hidden Layer 1 – 32 neurons and ReLU activation
- Hidden Layer 2 – 16 neurons and ReLU activation
- Output Layer – 1 neuron and sigmoid activation

2.2. Model Training Configuration

- Optimizer – Adam
- Loss – Binary Cross Entropy
- Epochs – 500
- Batch Size – 32

```
44/44 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step

Neural Network Model Report:
              precision    recall  f1-score   support

           0       0.83      0.82      0.82      1033
           1       0.52      0.52      0.52       374

    accuracy                           0.74      1407
   macro avg       0.67      0.67      0.67      1407
weighted avg       0.74      0.74      0.74      1407


Neural Network Model Confusion Matrix:
[[850 183]
 [179 195]]
```

Figure 11 - Neural Network Classifier Model Report

3. Model 2 – Decision Tree Classifier Using Scikit Learn

3.1. Model Configuration

- Criterion – Gini
- Max Depth – 6

```
Best Parameters:  {'criterion': 'gini', 'max_depth': 6, 'min_samples_split': 10}

Decision Tree Model Report:
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      1033
           1       0.61      0.53      0.57       374

    accuracy                           0.79      1407
   macro avg       0.72      0.70      0.71      1407
weighted avg       0.78      0.79      0.78      1407


Confusion Matrix:
[[907 126]
 [176 198]]
```

Figure 12 - Decision Tree Classifier Model Report

4. Model Comparison

- Neural Network Model – 74% Accuracy
- Decision Tree Model – 78% Accuracy

```
from sklearn.metrics import accuracy_score

nn_acc = accuracy_score(y_test, nn_predictions)
dt_acc = accuracy_score(y_test, dt_predictions)

print("Neural Network (TensorFlow) Model Accuracy:", nn_acc)
print("Decision Tree Model Accuracy:", dt_acc)

if nn_acc > dt_acc:
    print("\nNeural Network performs better overall")
else:
    print("\nDecision Tree performs better overall")
```

```
Neural Network (TensorFlow) Model Accuracy: 0.7427149964463398
Decision Tree Model Accuracy: 0.7853589196872779

Decision Tree performs better overall
```

Figure 13 - Model Comparison

Task 3 – AI Ethics and Post Deployment Strategies

1. Ethical Strategies in Model Development

- **Fairness and Bias Reduction**
  - Balanced stratified splitting prevents bias towards majority class
  - Encodings do not elevate or suppress sensitive attributes
- **Transparency**
  - Decision Trees provide explainable insights
  - EDA visuals help stakeholders understand churn factors
- **Responsible Data Handling**
  - No personally identifiable fields are used
  - Secure processing of billing and contract data

2. Post Deployment Ethical Strategy

- **Model Monitoring**
  - Monthly performance tracking
  - Detect accuracy drift or bias re-emergence
- **Retraining Policy**
  - Retrain if accuracy drops in the long term
- **Legal Compliance**
  - Follow a GDPR like guidelines for customer data
  - Avoid storing sensitive data unnecessarily
- **Explainability**
  - A simple dashboard showing the risk factors
- **Impact on Society and Customers**
  - Ensure that predictions are not used to unfairly deny services
  - Use churn prediction for customer retention benefits

3. Limitations and Future Enhancements

a. Limitations

- Dataset is moderately imbalanced
- Neural Network Model and Decision Tree Models have a fairly similar rate in accuracy (74% and 78% respectively), but according to this test at least, the Decision Tree Classifier has an accuracy upper hand by 4%.

b. Future Improvements

- Use other ML techniques such as XGBoost or Random Forest for boosted performance
- Apply SMOTE for better class balancing
- Deploy model through could API for easy updatability

Conclusion

The project successfully explored, preprocessed, modelled and evaluated the Telco Customer Churn dataset using two ML techniques, where the Decision Tree Model outperformed the Neural Network Model in predicting accuracy.

All implementation files are available in the following Git repository:
https://github.com/gowrisankar393/telco-customer-churn-prediction-model-2507578

An appendix section is included in this report, which contains the complete source code.

References

CampusX (2022). *Customer Churn Prediction using ANN | Keras and Tensorflow | Deep Learning Classification*. [online] YouTube. Available at: https://www.youtube.com/watch?v=9wmImImmgcI

Churn Prediction Dataset (Telco Customer Churn). IBM Sample Data, 2017. Available at: https://www.kaggle.com/blastchar/telco-customer-churn

codebasics (2020). *Customer churn prediction using ANN | Deep Learning Tutorial 18 (Tensorflow2.0, Keras & Python)*. [online] YouTube. Available at: https://www.youtube.com/watch?v=MSBY28IJ47U

Kingma, D.P. & Ba, J.L., 2015. *Adam: A Method for Stochastic Optimization*. International Conference on Learning Representations (ICLR). Available at: https://arxiv.org/abs/1412.6980

Scikit-Learn Developers, 2024. *Scikit-Learn Documentation: Decision Trees*. Available at: https://scikit-learn.org/stable/modules/tree.html

Siddhardhan (2024). *Project 22. Customer Churn Prediction Using Machine Learning | Complete ML Project Walkthrough* 🚀. [online] YouTube. Available at: https://www.youtube.com/watch?v=qNglJgNOb7A

TensorFlow Developers, 2024. *TensorFlow API Documentation*. Available at: https://www.tensorflow.org/api_docs

www.youtube.com. (n.d.). *Decision Tree Regression Clearly Explained!* [online] Available at: https://www.youtube.com/watch?v=UhY5vPfQIrA.

www.youtube.com. (n.d.). *How to implement Decision Trees from scratch with Python*. [online] Available at: https://www.youtube.com/watch?v=NxEHSAfFlK8

Appendix

1. Import Dataset and Load Dataset

```
from google.colab import files
import pandas as pd

print("Please upload the Telco Customer Churn CSV file")
uploaded = files.upload()

#read the uploaded file automatically
file_name = list(uploaded.keys())[0]
df = pd.read_csv(file_name)

print("\nUpload successful. First few rows of the dataset:")
df.head()
```

2. Task 1 - Exploratory Data Analysis

2.1. Dataset Info

```
print("Dataframe Shape:", df.shape)

print("\nDataset Info")
df.info()

print("\n\nMissing Values in Each Column", df.isnull().sum(),"\n\n")

#categorical summary of the dataset
print("Categorical Summary:")
df.describe(include='all')
for col in df.select_dtypes(include='object'):
    print(f"\nValue counts for {col}:")
    print(df[col].value_counts())
```

2.2. Data Cleaning

```
#convert TotalCharges to numeric values as some fields contain empty
strings
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
print('Empty fields converted to numeric values')

#remove rows where TotalCharges is NaN (not a huge probelm as only a very
few are there)
df = df.dropna(subset=['TotalCharges'])
print('Removed rows where TotalCharges is NaN')

#reset index after deletion
df.reset_index(drop=True, inplace=True)

#cleaned dataset shape
print("Dataset shape after cleaning:", df.shape)
```

2.3. Visual Exploratory Data Analysis

```
import seaborn as sns
import matplotlib.pyplot as plt

#churn distribution
plt.figure(figsize=(6,4))
sns.countplot(data=df, x='Churn')
plt.title("Churn Distribution")
plt.show()

#monthly charges distribution
plt.figure(figsize=(6,4))
sns.histplot(df['MonthlyCharges'], kde=True)
plt.title("Monthly Charges Distribution")
plt.show()

#contract type effect on churn
```

```
plt.figure(figsize=(7,5))
sns.countplot(data=df, x='Contract', hue='Churn')
plt.title("Churn Rate by Contract Type")
plt.show()

#correlation heatmap
plt.figure(figsize=(10,7))
sns.heatmap(df.select_dtypes(include='number').corr(), annot=True,
cmap="coolwarm")
plt.title("Correlation Heatmap (Numeric Features)")
plt.show()
```

3. Task 2 – Data Preprocessing and Model Implementation

3.1. Data Preprocessing

```
from sklearn.preprocessing import LabelEncoder, StandardScaler

df_processed = df.copy()

#removing customerID for this process as its a unique identofier and not a
feature
df_processed=df_processed.drop(['customerID'], axis=1)

#encode yes/no columns and making them binary values
binary_cols = ['Partner', 'Dependents', 'PhoneService',
'PaperlessBilling', 'Churn']
for col in binary_cols:
    df_processed[col] = df_processed[col].map({'Yes': 1, 'No': 0})

#one hot encode multi category categorical columns
df_processed = pd.get_dummies(df_processed, drop_first=True)

#separate features and labels
X = df_processed.drop('Churn', axis=1)
```

```
y = df_processed['Churn']

#scale numerical data for neural network model
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Data Preprocessing completed!")
print("Number of features:", X_scaled.shape[1])
```

3.2. Splitting Data to Training and test Sets

```
from sklearn.model_selection import train_test_split

#spltiing data to 80:20 ratio for training and testing
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y)

print("Training samples:", len(X_train))
print("Testing samples:", len(X_test))
```

3.3. Neural Network Model

```
import tensorflow as tf
from tensorflow.keras import layers, models

#neural network model
nn_model = models.Sequential()
nn_model.add(layers.Dense(32, activation='relu'))
nn_model.add(layers.Dense(16, activation='relu'))
nn_model.add(layers.Dense(1, activation='sigmoid'))#for binary
classification

nn_model.compile(optimizer='adam',
                 loss='binary_crossentropy',
                 metrics=['accuracy'])
```

```
#train the model
history = nn_model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=500,
    batch_size=32,
    verbose=1
)

#print a confirmation
print("\nNeural Network Model Training Complete")
```

3.4. Model Evaluation

```
from sklearn.metrics import classification_report, confusion_matrix

nn_predictions = (nn_model.predict(X_test) > 0.5).astype("int32")

print("\nNeural Network Model Report:")
print(classification_report(y_test, nn_predictions))

print("\nNeural Network Model Confusion Matrix:")
print(confusion_matrix(y_test, nn_predictions))
```

3.5. Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

#parameter grid for tuning
param_grid={
    'max_depth':[2, 4, 6, 8, 10],
    'criterion':['gini','entropy'],
    'min_samples_split':[2, 5, 10]
}
```

```
#train Decision Tree Model with hyper paramenter tuning
grid_dt=GridSearchCV(estimator=DecisionTreeClassifier(),
param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)

grid_dt.fit(X_train, y_train)

print("Best Parameters: ", grid_dt.best_params_)

best_dt=grid_dt.best_estimator_

dt_predictions = best_dt.predict(X_test)

print("\nDecision Tree Model Report:")
print(classification_report(y_test, dt_predictions))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, dt_predictions))
```

3.6. Model Comparison

```
from sklearn.metrics import accuracy_score

nn_acc = accuracy_score(y_test, nn_predictions)
dt_acc = accuracy_score(y_test, dt_predictions)

print("Neural Network (TensorFlow) Model Accuracy:", nn_acc)
print("Decision Tree Model Accuracy:", dt_acc)

if nn_acc > dt_acc:
    print("\nNeural Network performs better overall")
else:
    print("\nDecision Tree performs better overall")
```