6.3 Document your entire process, findings, and recommendations in a well structured report.

Telecom Customer Churn Analysis Report
Prepared by Darshan Rana
Date: 2/9/23

1. Data Exploration and Preprocessing

In this section, we loaded the telecom dataset and performed preliminary data exploration and preprocessing.

Code:

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv("telecom.csv")

# Basic information about the dataset
df.info()

# Checking for missing values
df.isnull().sum()

# Data Visualization of Churn Distribution
plt.figure(figsize=(8, 6))
sns.countplot(df['Churn'])
plt.title("Distribution of Churn")
plt.xlabel("Churn")
plt.ylabel("Count")
plt.show()
```

2. Feature Engineering

In this section, we conducted feature engineering to prepare the data for modeling. Key steps included:

- Encoding the target variable "Churn" as binary (0 for "No" and 1 for "Yes").
- Creating dummy variables for categorical features using one-hot encoding.

Code:

```
# Encode the target variable
df['Churn'].replace(to_replace='Yes', value=1, inplace=True)
df['Churn'].replace(to_replace='No',  value=0, inplace=True)

# Convert categorical variables into dummy variables
df_dummies = pd.get_dummies(df)
```

3. Model Building

In this section, we built machine learning models for predicting customer churn. We trained and evaluated three different models: Decision Tree, K Nearest Neighbors (KNN), and Random Forest. Here is a summary of the model building process:

Code:

```
# Split the dataset into training and testing sets
from sklearn.model_selection import train_test_split

X = df_dummies.drop('Churn', axis=1)
y = df_dummies['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

df_dummies_classifier = DecisionTreeClassifier(random_state=42)
df_dummies_classifier.fit(X_train, y_train)
```

```
# Hyperparameter tuning using Grid Search
from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth': [3, 5, 7]}
grid_search = GridSearchCV(df_dummies_classifier, param_grid, cv=5)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_

# Accuracy
y_pred = df_dummies_classifier.predict(X_test)
df_accuracy = accuracy_score(y_test, y_pred)

# K Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier

knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(X_train, y_train)

# Accuracy
knn_pred = knn_classifier.predict(X_test)
knn_accuracy = accuracy_score(y_test, knn_pred)

# Neural Network using TensorFlow and Keras
import tensorflow as tf
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=32)
nn_loss, nn_accuracy = model.evaluate(X_test, y_test)

# Random Forest
from sklearn.ensemble import RandomForestClassifier

rf_classifier = RandomForestClassifier(random_state=42)
```

```
rf_classifier.fit(X_train, y_train)

# Accuracy
rf_pred = rf_classifier.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
```

4. Model Evaluation

In this section, we evaluated the performance of each model using various metrics such as accuracy, precision, recall, and F1-score. We also compared the models to determine the best-performing one.

Code:

```
# Function to evaluate a model and return evaluation metrics
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    return accuracy, precision, recall, f1

# Evaluate the Decision Tree classifier
accuracy_df, precision_df, recall_df, f1_df = evaluate_model(df_dummies_classifier,
X_test, y_test)

# Evaluate the K Nearest Neighbors (KNN) classifier
accuracy_knn, precision_knn, recall_knn, f1_knn = evaluate_model(knn_classifier,
X_test, y_test)

# Evaluate the Random Forest classifier
accuracy_rf, precision_rf, recall_rf, f1_rf = evaluate_model(rf_classifier

, X_test, y_test)

# Evaluate the Neural Network (NN) model
accuracy_nn, precision_nn, recall_nn, f1_nn = evaluate_model(model, X_test, y_test)
```

```
# Create a summary table
evaluation_summary = pd.DataFrame({
    'Model': ['Decision Tree', 'K Nearest Neighbors', 'Random Forest', 'Neural Network'],
    'Accuracy': [accuracy_df, accuracy_knn, accuracy_rf, accuracy_nn],
    'Precision': [precision_df, precision_knn, precision_rf, precision_nn],
    'Recall': [recall_df, recall_knn, recall_rf, recall_nn],
    'F1-Score': [f1_df, f1_knn, f1_rf, f1_nn]
})
```

5. Conclusion

In this report, we conducted an analysis of telecom customer churn using machine learning models. We explored the data, performed preprocessing and feature engineering, built and evaluated several models, and summarized the results in a table. Here are the key takeaways:

- The Random Forest model achieved the highest accuracy and F1-Score, making it the best-performing model for predicting customer churn in this dataset.
- Other models, such as Decision Tree, K Nearest Neighbors, and Neural Network, also provided competitive results.
- Further analysis and feature engineering could potentially improve the predictive performance of the models.