1. Importing Necessary Libraries

The script starts by importing the required libraries for data handling, visualization, preprocessing, machine learning, deep learning, and clustering.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier, export_text
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.cluster import KMeans
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv1D, Flatten, Dropout
```

- **Pandas & NumPy**: For handling and manipulating datasets.
- **Matplotlib & Seaborn**: For data visualization.
- **Scikit-learn**: For data preprocessing, model training, and evaluation.
- **TensorFlow/Keras**: For building deep learning models.
- **KMeans**: For clustering analysis.

2. Loading Datasets :

```
raw_data = pd.read_csv("telecom_customer_churn_dataset.csv")
binary_data = pd.read_csv("telecom_customer_churn_dataset_binary.csv")

df1 = pd.read_csv("telecom_customer_churn_dataset.csv")
df2 = pd.read_csv("telecom_customer_churn_dataset_binary.csv")
```

The script loads two datasets:

- `raw_data`: Main dataset containing customer details.
- `binary_data`: A modified dataset that may have binary-encoded categorical values.

3. Exploratory Data Analysis (EDA)

Checking for Missing Values

python
CopyEdit

```
print("Checking for missing values:\n", raw_data.isnull().sum())
```

This checks for missing values in the dataset.

Summary Statistics

```
print("\nDataset Summary:\n", raw_data.describe())
```

- Provides statistical summaries such as mean, min, max, etc.

Encoding Categorical Variables

```
le = LabelEncoder()for col in
raw_data.select_dtypes(include=['object']).columns:
    if col == "Churn":
        continue
    raw_data[col] = le.fit_transform(raw_data[col])
```

- Converts categorical features into numerical values using `LabelEncoder`.

Correlation Heatmap

```
plt.figure(figsize=(12, 6))
sns.heatmap(raw_data.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```

- Displays correlations between numerical features.

Churn Distribution

```
sns.countplot(x="Churn Label", data=df1)
plt.title("Churn Distribution")
plt.show()
```

- Plots the distribution of churned vs. non-churned customers.

4. Data Preprocessing

Defining Features and Target Variable

```
X = raw_data.drop(columns=["Churn Label"])
y = raw_data["Churn Label"]
```

- `X`: Independent variables (features).
- `y`: Dependent variable (churn label).

Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

- Splits the dataset into **80% training** and **20% testing**.

Feature Scaling

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- Standardizes features to improve model performance.

5. Classification Models

1. Naïve Bayes Classifier

```
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
y_pred_nb = nb_model.predict(X_test)print("\nNaïve Bayes Accuracy:",
accuracy_score(y_test, y_pred_nb))print(classification_report(y_test,
y_pred_nb))
```

- Implements **Naïve Bayes**, a probabilistic classifier.

2. Decision Tree Classifier

```
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)print("\nDecision Tree Accuracy:",
accuracy_score(y_test, y_pred_dt))print(classification_report(y_test,
y_pred_dt))
```

- Implements a **Decision Tree**, a rule-based classifier.

6. Deep Learning Models

3. Convolutional Neural Network (CNN)

```
X_train_cnn = np.expand_dims(X_train, axis=2)
X_test_cnn = np.expand_dims(X_test, axis=2)

cnn_model = Sequential([
    Conv1D(filters=64, kernel_size=2, activation="relu",
input_shape=(X_train_cnn.shape[1], 1)),
    Dropout(0.3),
    Flatten(),
    Dense(64, activation="relu"),
    Dense(1, activation="sigmoid")
])

cnn_model.compile(optimizer="adam", loss="binary_crossentropy",
metrics=["accuracy"])
cnn_model.fit(X_train_cnn, y_train, epochs=10, batch_size=32,
validation_data=(X_test_cnn, y_test))
cnn_pred = (cnn_model.predict(X_test_cnn) >
0.5).astype("int32")print("\nCNN Accuracy:", accuracy_score(y_test,
cnn_pred))
```

- Uses **CNN** for churn prediction.

4. Fully Connected Neural Network (Deep Learning Model)

```
dl_model = Sequential([
    Dense(128, activation="relu", input_shape=(X_train.shape[1],)),
    Dense(64, activation="relu"),
    Dense(32, activation="relu"),
    Dense(1, activation="sigmoid")
])

dl_model.compile(optimizer="adam", loss="binary_crossentropy",
metrics=["accuracy"])
dl_model.fit(X_train, y_train, epochs=20, batch_size=32,
validation_data=(X_test, y_test))
dl_pred = (dl_model.predict(X_test) >
0.5).astype("int32")print("\nDeep Learning Model Accuracy:",
accuracy_score(y_test, dl_pred))
```

- Implements a **deep learning model with multiple dense layers**.

7. Decision Tree for Binary Dataset

```
X_bin = binary_data.drop(columns=["Churn Label"])
y_bin = binary_data["Churn Label"]
for col in X_bin.select_dtypes(include=['object']).columns:
    X_bin[col] = le.fit_transform(X_bin[col])

dt_bin_model = DecisionTreeClassifier(random_state=42)
dt_bin_model.fit(X_bin, y_bin)

tree_rules = export_text(dt_bin_model,
feature_names=list(X_bin.columns))print("\nDecision Tree Rules:\n",
tree_rules)
```

- Builds a **decision tree** for the binary dataset.

8. K-Means Clustering

```
X_cluster = raw_data.drop(columns=["Churn Label"])
kmeans = KMeans(n_clusters=3, random_state=42)
raw_data["Cluster"] = kmeans.fit_predict(X_cluster)

plt.figure(figsize=(8, 5))
sns.scatterplot(x=raw_data["Tenure"], y=raw_data["Monthly Charges"],
hue=raw_data["Cluster"], palette="viridis")
plt.title("Tenure vs Monthly Charges Clustering")
plt.show()
```

- Applies **K-Means clustering** to segment customers.

Summary

This script:

1. Loads and processes the dataset.
2. Performs **EDA** and visualizations.
3. Prepares data for **classification** and **deep learning** models.
4. Implements:
    1. **Naïve Bayes**
    2. **Decision Tree**
    3. **CNN**
    4. **Fully Connected Neural Network**
5. Uses **K-Means Clustering** to segment customers.
6. Trains a **Decision Tree** to analyze causes of churn.