ASS-1

```
1) import pandas as pd
2) import numpy as np
3) import matplotlib.pyplot as plt
4) import seaborn as sns
5) from sklearn.model_selection import train_test_split
6) from sklearn.pipeline import make_pipeline
7) from sklearn.preprocessing import StandardScaler
8) from sklearn.linear_model import LinearRegression
9) from sklearn.ensemble import RandomForestRegressor
10) from sklearn.metrics import r2_score, mean_squared_error
11) df = pd.read_csv("C:\\Users\\Vijay\\DSBDA\\uber.csv")
12) df.head()
13) df.isna().sum()
14) df.boxplot()
15) plt.show()
16) df.dropna(inplace=True)
17) df.drop('Unnamed: 0', axis=1, inplace=True)
18) df.drop('pickup_datetime', axis=1, inplace=True)
19) numeric_df = df.select_dtypes(include=['number'])
20) correlation = numeric_df.corr()
21) plt.figure(figsize=(8, 6))
22) sns.heatmap(correlation, annot=True, cmap='coolwarm')
23) plt.title("Correlation Matrix")
24) plt.show()
25) X = df.drop(['fare_amount', 'key'], axis=1)
26) Y = df['fare_amount']
27) X_train, X_test, y_train, y_test = train_test_split(X,
 Y, test_size=0.2, random_state=42)
28) model = make_pipeline(StandardScaler
(with_mean=False), LinearRegression())
29) model.fit(X_train, y_train)
30) y_pred = model.predict(X_test)
31) plt.figure(figsize=(8, 6))
32) plt.scatter(y_test, y_pred, alpha=0.5, color='blue')
33) plt.plot([y_test.min(), y_test.max()],
[y_test.min(), y_test.max()], color='red')
34) plt.xlabel("Actual Fare Amount")
35) plt.ylabel("Predicted Fare Amount")
36) plt.title("Linear Regression: Actual vs Predicted Fares")
37) plt.grid(True)
```

```
plt.show()
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
rf = RandomForestRegressor(n_estimators=10,
random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
r2_lr = r2_score(y_test, y_pred_lr)
rmse_lr = np.sqrt(mean_squared_error
(y_test, y_pred_lr))
r2_rf = r2_score(y_test, y_pred_rf)
rmse_rf = np.sqrt(mean_squared_error
(y_test, y_pred_rf))
print("Linear Regression:")
print(f"R2 Score: {r2_lr:.4f}")
print(f"RMSE: {rmse_lr:.4f}")
print("\nRandom Forest Regression:")
print(f"R2 Score: {r2_rf:.4f}")
print(f"RMSE: {rmse_rf:.4f}")
```

ASS-2

```
1) import pandas as pd
2) df = pd.read_csv("/home/comp/Documents/emails.csv")
3) df
4) df.columns
5) df.shape
6) x = df.drop(['Email No.','Prediction'], axis = 1)
7) y = df['Prediction']
8) x.dtypes
9) y.value_counts
10) import seaborn as sns
11) sns.countplot(x=y)
12) from sklearn.preprocessing import MinMaxScaler
13) scaler = MinMaxScaler()
14) x_scaled = scaler.fit_transform(x)
15) from sklearn.model_selection import train_test_split
16) x_train, x_test, y_train, y_test = train_test_split
(x_scaled, y, random_state=0, test_size=0.25)
17) from sklearn.neighbors import KNeighborsClassifier
18) model = KNeighborsClassifier(n_neighbors=5)
19) model.fit(x_train, y_train)
20) y_pred = model.predict(x_test)
21) from sklearn.metrics import ConfusionMatrixDisplay,
accuracy_score, classification_report
22) ConfusionMatrixDisplay.from_predictions
(y_test, y_pred)
23) print(classification_report(y_test, y_pred))
24) accuracy_score = (y_test, y_pred)
25) import numpy as np
26) import matplotlib.pyplot as plt
27) error = []
28) for k in range(1,41):
29)     model = KNeighborsClassifier(n_neighbors=k)
30)     model.fit(x_train, y_train)
31)     pred = model.predict(x_test)
32)     error.append(np.mean(pred != y_test))
33) plt.figure(figsize=(16,9))
34) plt.xticks(range(1,41))
35) plt.grid()
36) plt.xlabel("value of predictions")
37) plt.ylabel("value of error")
```

38) plt.plot(range(1,41), error, marker='.')
39) from sklearn.svm import SVC
40) model = SVC(kernel='linear')
41) model.fit(x_train, y_train)
42) y_pred = model.predict(x_test)
43) from sklearn.metrics import accuracy_score
44) accuracy_score(y_test, y_pred)
45) print(classification_report(y_test, y_pred))
46) model = KNeighborsClassifier(n_neighbors=1)
47) model.fit(x_train, y_train)
48) y_pred = model.predict(x_test)
49) accuracy_score(y_test, y_pred)

ASS-3

```
1) import matplotlib.pyplot as plt
2) import seaborn as sns
3) import pandas as pd
4) import numpy as np
5) data = pd.read_csv("/home/comp/Downloads/Churn_Modelling.csv")
6) data.head()
7) X = data.drop(['Exited'], axis=1)
8) y = data['Exited']
9) X = pd.get_dummies(X, drop_first=True)
10) X.fillna(X.mean(), inplace=True)
11) from sklearn.model_selection import train_test_split
12) X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
13) from sklearn.preprocessing import StandardScaler
14) scaler = StandardScaler()
15) X_train_scaled = scaler.fit_transform(X_train)
16) X_test_scaled = scaler.transform(X_test)
17) from tensorflow.keras.models import Sequential
18) from tensorflow.keras.layers import Dense
19) model = Sequential()
20) model.add(Dense(units=64, activation='relu',
input_dim=X_train_scaled.shape[1]))
21) model.add(Dense(units=32, activation='relu'))
22) model.add(Dense(units=1, activation='sigmoid'))
23) model.compile(optimizer='adam', loss=
'binary_crossentropy', metrics=['accuracy'])
24) model.summary()
25) history = model.fit(X_train_scaled, y_train, validation_data
=(X_test_scaled, y_test), epochs=50, batch_size=32)
26) from sklearn.metrics import accuracy_score,
confusion_matrix
27) plt.figure(figsize=(12, 5))
28) plt.subplot(1, 2, 1)
29) plt.plot(history.history['accuracy'],
 label='Training Accuracy')
30) plt.plot(history.history['val_accuracy'],
label='Validation Accuracy')
31) plt.title('Training and Validation Accuracy')
32) plt.xlabel('Epochs')
33) plt.ylabel('Accuracy')
```

```
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'],
label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()
y_pred = (model.predict(X_test_scaled)
> 0.5).astype('int32')
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues', xticklabels=['Stayed', 'Exited'],
yticklabels=['Stayed', 'Exited'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

ASS-4

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import seaborn as sns
data = pd.read_csv('C:\\Users\\Vijay\\Downloads
\\sales_data_sample.csv', encoding='latin1')
print("Dataset Preview:")
print(data.head())
cols_to_remove = ['ORDERNUMBER', 'ORDERDATE',
'CUSTOMERNAME', 'PHONE', 'ADDRESSLINE1',
 'ADDRESSLINE2', 'CITY', 'STATE', 'POSTALCODE',
'COUNTRY', 'TERRITORY', 'CONTACTLASTNAME',
 'CONTACTFIRSTNAME', 'DEALSIZE']
data_numeric = data.drop(columns=
cols_to_remove, errors='ignore')
data_numeric = data_numeric.dropna()
data_numeric = data_numeric
.select_dtypes(include=[np.number])
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_numeric)
inertia = []
K = range(1, 11)
for k in K:
    kmeans = KMeans(n_clusters=k,
    init='k-means++', random_state=42)
    kmeans.fit(data_scaled)
    inertia.append(kmeans.inertia_)
plt.figure(figsize=(8, 5))
plt.plot(K, inertia, 'bo-', markersize=8)
plt.title('Elbow Method to Determine Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia (Within-cluster Sum of Squares)')
plt.show()
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k,
init='k-means++', random_state=42)
data_numeric['Cluster'] =
 kmeans.fit_predict(data_scaled)
```

```
print("\nCluster Centers (scaled):")
print(kmeans.cluster_centers_)
print("\nCluster Distribution:")
print(data_numeric['Cluster'].value_counts())
plt.figure(figsize=(8, 6))
sns.scatterplot(x=data_scaled[:, 0],
y=data_scaled[:, 1],
 hue=data_numeric['Cluster'], palette='viridis', s=60)
plt.title('K-Means Clustering Visualization')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```