INDEX

SNO	DATE	EXPERIMENT NAME	PAGE NO	MARKS	SIGNATURE
Additional Experiment					
Model lab					
			Average:		

NAME: ___________________

REG NO: __________________

Ex.No : Date:	**Data Pre-processing & EDA using Pandas and Matplotlib.**

AIM:

ALGORITHM:

PROGRAM

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Step 1: Given dataset (with missing values added)

data = {
```

```python
 'Name': ['Jai','Princi','Ganga','Anuj','Ravi','Natasha','Riya'],
 'Age': ['17','18',np.nan,'19','18','17',np.nan],
 'Gender': ['M','F','F','M','M','F','F'],
 'Marks': ['90','76','86',np.nan,'82','88','67']
}

df = pd.DataFrame(data)
print("Original Dataset:\n")
print(df)

# Step 2: Data type conversion
df['Age'] = pd.to_numeric(df['Age'])
df['Marks'] = pd.to_numeric(df['Marks'])

# Step 3: Handling missing values
df['Age'].fillna(df['Age'].mean(), inplace=True)
df['Marks'].fillna(df['Marks'].mean(), inplace=True)
print("\nAfter Handling Missing Values:\n")
print(df)

# Step 4: Encoding categorical variables
df_encoded = pd.get_dummies(df, columns=['Gender'], drop_first=True)
print("\nAfter Encoding Categorical Variables:\n")
print(df_encoded)

# Step 5: Normalization (Min-Max Scaling)
df_encoded['Age'] = (df_encoded['Age'] - df_encoded['Age'].min()) / \
(df_encoded['Age'].max() - df_encoded['Age'].min())
df_encoded['Marks'] = (df_encoded['Marks'] - df_encoded['Marks'].min()) / \
(df_encoded['Marks'].max() - df_encoded['Marks'].min())
print("\nAfter Normalization:\n")
print(df_encoded)

# Step 6: Data Visualization
```

```python
# Bar chart for Marks
plt.bar(df['Name'], df['Marks'])
plt.xlabel("Student Name")
plt.ylabel("Marks")
plt.title("Marks of Students")
plt.show()
# Pie chart for Gender distribution
df['Gender'].value_counts().plot.pie(autopct='%1.1f%%')
plt.title("Gender Distribution")
plt.ylabel("")
plt.show()
```

OUTPUT

Department of Computer Science and Engineering		
Preparation	25	
Performance	25	
Record	40	
Viva	10	
Total	100	
Signature		

RESULT:

NAME: ___________________

REG NO: _________________

Ex.No : Date:	**Linear Regression and Model Evaluation using MAE, MSE, R^2, and plot residuals**

AIM:

ALGORITHM:

PROGRAM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Step 1: Create dataset (with error)
data = {
'Hours_Studied': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
'Marks_Scored': [38, 41, 50, 52, 58, 63, 65, 71, 76, 82]  # noisy data
}
df = pd.DataFrame(data)
print(df)
# Step 2: Define independent and dependent variables
X = df[['Hours_Studied']]
y = df['Marks_Scored']

# Step 3: Split dataset
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=1

# Step 4: Train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 5: Predict
y_pred = model.predict(X_test)

# Step 6: Evaluation metrics
print("\nEvaluation Metrics:")
print("MAE:", mean_absolute_error(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
print("R² :", r2_score(y_test, y_pred))

# Step 7: Residual plot
residuals = y_test - y_pred
plt.scatter(y_pred, residuals)
```

```
plt.axhline(0)
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.title("Residual Plot (With Error Data)")
plt.show()
```

OUTPUT

Department of Computer Science and Engineering		
Preparation	25	
Performance	25	
Record	40	
Viva	10	
Total	100	
Signature		

RESULT:

NAME: ___________________

REG NO: __________________

Ex.No : Date:	**Logistic Regression for Classification**

AIM:

ALGORITHM:

PROGRAM

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import roc_curve, auc, ConfusionMatrixDisplay
```

```
#Step 1: Dataset
X = np.array([29, 15, 33, 28, 39]).reshape(-1, 1)          # Hours studied
y = np.array([0, 0, 1, 1, 1])                               # Pass(1)/Fail(0)

#Step 2: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=42
)

# Step 3: Train Logistic Regression Model
model = LogisticRegression()
model.fit(X_train, y_train)

#Step 4: Predictions
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]

#Step 5: Confusion Matrix & Accuracy
m = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix")
plt.show()
print("Accuracy:", accuracy_score(y_test, y_pred))

#Step 6: ROC Curve & AUC
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label="ROC Curve (AUC = %.2f)" % roc_auc)
plt.plot([0, 1], [0, 1], linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
```

```
plt.legend()

plt.show()

#Step 7: Probability for Given Hours (33)

prob_33 = model.predict_proba([[33]])[0][1]

print("Probability of passing with 33 hours study:", prob_33)

# Step 8: Minimum Hours for >95% Probability

for hours in range(0, 61):

prob = model.predict_proba([[hours]])[0][1]

if prob > 0.95:

print("Minimum hours required for >95% pass probability:", hours)

print("Probability at this hour:", prob)

break
```

OUTPUT

Department of Computer Science and Engineering		
Preparation	25	
Performance	25	
Record	40	
Viva	10	
Total	100	
Signature		

RESULT:

NAME: ___________________

REG NO: _________________

Ex.No : Date:	**K-Nearest Neighbors (K-NN) Classifier**

AIM:

ALGORITHM:

PROGRAM

```
import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

# ---------------------------------

# Step 1: Dataset (Height, Weight)

# ---------------------------------
```

```python
X = np.array([[167, 51],[182, 62],[176, 69],[173, 64],[172, 65],[174, 56],[169, 58],
[173, 57],[170, 55]])
y = np.array(["Underweight","Normal","Normal","Normal","Normal","Underweight",
"Normal","Normal","Normal"])
# New data point to classify
new_point = np.array([[170, 57]])

# Step 2: Normalization
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
new_point_scaled = scaler.transform(new_point)

# Step 3: Apply k-NN (Euclidean Distance)
k = 3   # You can change k value
model = KNeighborsClassifier(n_neighbors=k, metric='euclidean')
model.fit(X_scaled, y)
# Prediction
prediction = model.predict(new_point_scaled)
print("Predicted Class for (170 cm, 57 kg):", prediction[0])
```

OUTPUT

Department of Computer Science and Engineering		
Preparation	25	
Performance	25	
Record	40	
Viva	10	
Total	100	
Signature		

RESULT: