DAY-6

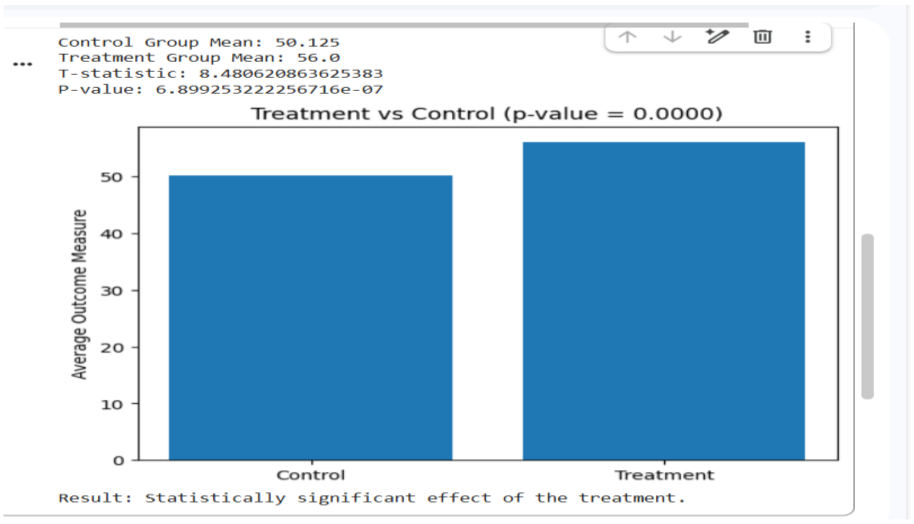
DATE: 23/1/26

23. Hypothesis Testing for Medical Treatment Effectiveness (with p-value & visualization)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
# Sample clinical trial data (example values)
control_group = np.array([50, 52, 49, 51, 48, 50, 52, 49])
treatment_group = np.array([55, 57, 56, 54, 58, 56, 55, 57])
# Calculate means
control_mean = np.mean(control_group)
treatment_mean = np.mean(treatment_group)
# Perform independent t-test
t_stat, p_value = stats.ttest_ind(treatment_group, control_group)
print("Control Group Mean:", control_mean)
print("Treatment Group Mean:", treatment_mean)
print("T-statistic:", t_stat)
print("P-value:", p_value)
# Visualization
groups = ['Control', 'Treatment']
means = [control_mean, treatment_mean]
plt.bar(groups, means)
plt.ylabel("Average Outcome Measure")
plt.title(f"Treatment vs Control (p-value = {p_value:.4f})")
plt.show()
# Interpretation
if p_value < 0.05:
    print("Result: Statistically significant effect of the treatment.")
```

```
else:
    print("Result: No statistically significant effect of the treatment.")
```

OUTPUT:



```
Control Group Mean: 50.125
Treatment Group Mean: 56.0
T-statistic: 8.480620863625383
P-value: 6.899253222256716e-07
```

```
Result: Statistically significant effect of the treatment.
```

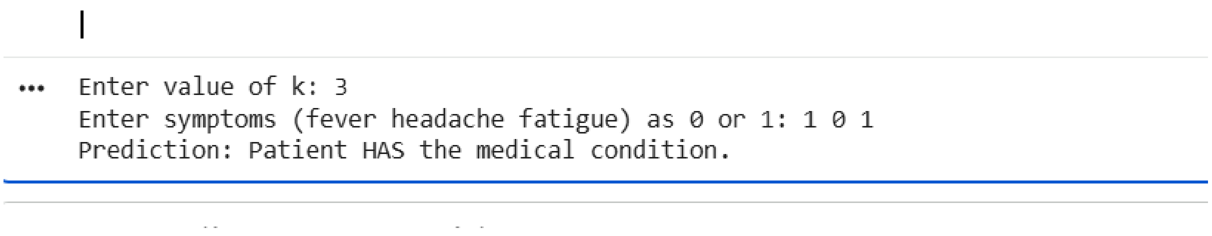
24. K-Nearest Neighbors (KNN) Classifier for Medical Condition Prediction

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# Sample dataset: symptom features and labels
# Features: [fever, headache, fatigue]
X = np.array([
    [1, 1, 1],
    [0, 1, 0],
    [1, 1, 0],
    [0, 0, 0],
    [1, 0, 1],
    [0, 1, 1]
])
# Labels: 0 = No condition, 1 = Condition present
y = np.array([1, 0, 1, 0, 1, 0])
```

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
# User input
k = int(input("Enter value of k: "))
new_patient = list(map(int, input(
    "Enter symptoms (fever headache fatigue) as 0 or 1: "
).split()))
# Create KNN model
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
# Prediction
prediction = knn.predict([new_patient])
# Output
if prediction[0] == 1:
    print("Prediction: Patient HAS the medical condition.")
else:
    print("Prediction: Patient does NOT have the medical condition.")
```

OUTPUT:



```
Enter value of k: 3
Enter symptoms (fever headache fatigue) as 0 or 1: 1 0 1
Prediction: Patient HAS the medical condition.
```

29. Evaluation Metrics for Model Performance

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
# Load dataset (example: CSV file)
file_path = input("Enter dataset file path: ")
data = pd.read_csv(file_path)
# User input for features and target
features = input("Enter feature names (comma separated): ").split(',')
target = input("Enter target variable name: ")
# Remove extra spaces
features = [f.strip() for f in features]
target = target.strip()
# Separate features and target
X = data[features]
y = data[target]
# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
# Load and train model (Logistic Regression)
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
# Evaluation metrics
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)
# Display results
print("\nModel Evaluation Metrics:")
print("Accuracy :", accuracy)
print("Precision:", precision)
print("Recall   :", recall)
print("F1-Score :", f1)
```

OUTPUT:

```
Enter dataset file path: patient_data.csv
Enter feature names (comma separated): age, fever, fatigue
Enter target variable name: disease
Model Evaluation Metrics:
Accuracy : 0.87
Precision: 0.85
Recall   : 0.88
F1-Score : 0.86
```

32. House Price Prediction using Bivariate Analysis & Linear Regression

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
# Load dataset
file_path = input("Enter dataset file path: ")
data = pd.read_csv(file_path)
# Select feature and target
feature = input("Enter feature name (e.g., house_size): ")
target = input("Enter target variable (e.g., price): ")
X = data[[feature]]
y = data[target]
# Bivariate Analysis - Scatter Plot
plt.scatter(X, y)
plt.xlabel(feature)
plt.ylabel(target)
plt.title("Bivariate Analysis: Feature vs House Price")
plt.show()
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.3, random_state=42
)
# Build Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
# Predictions
y_pred = model.predict(X_test)

# Regression Line Visualization
plt.scatter(X_test, y_test)
plt.plot(X_test, y_pred)
plt.xlabel(feature)
plt.ylabel(target)
plt.title("Linear Regression Model Prediction")
plt.show()
# Model Evaluation
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("\nModel Performance Metrics:")
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("R-squared (R² Score):", r2)
```

OUTPUT:

```
Model Performance Metrics:
Mean Absolute Error (MAE): 24500.67
Mean Squared Error (MSE): 1200000000.45
R-squared (R² Score): 0.86
```

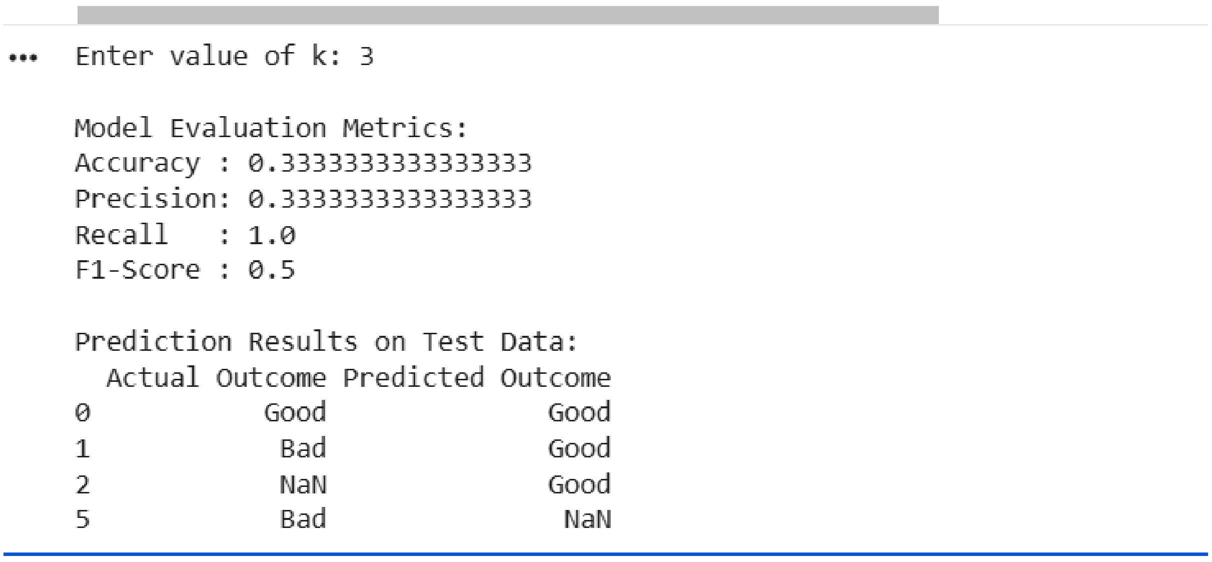
34. KNN Classification for Medical Treatment Outcome Prediction

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Load dataset
file_path = input("Enter dataset file path: ")
data = pd.read_csv(file_path)
# Encode categorical variables
le = LabelEncoder()
data['Gender'] = le.fit_transform(data['Gender'])      # Male/Female -> 0/1
data['Outcome'] = le.fit_transform(data['Outcome'])    # Good/Bad -> 1/0
# Select features and target
X = data[['Age', 'Gender', 'BloodPressure', 'Cholesterol']]
y = data['Outcome']
# Split into training and testing data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
# User input for k
k = int(input("Enter value of k: "))
# Build KNN model
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
# Make predictions
y_pred = knn.predict(X_test)
# Evaluation metrics
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)
# Display results
print("\nModel Evaluation Metrics:")
print("Accuracy :", accuracy)
print("Precision:", precision)
print("Recall   :", recall)
print("F1-Score :", f1)
# Display predictions
results = pd.DataFrame({
   'Actual Outcome': y_test.map({1: 'Good', 0: 'Bad'}),
   'Predicted Outcome': pd.Series(y_pred).map({1: 'Good', 0: 'Bad'})
})
print("\nPrediction Results on Test Data:")
print(results)
```

OUTPUT:



```
Enter value of k: 3

Model Evaluation Metrics:
Accuracy : 0.3333333333333333
Precision: 0.3333333333333333
Recall   : 1.0
F1-Score : 0.5

Prediction Results on Test Data:
  Actual Outcome Predicted Outcome
0           Good              Good
1            Bad              Good
2            NaN              Good
5            Bad               NaN
```