Name: Ujwal Sahu

Branch: Bvoc – AIDS

Division: B

EXPERIMENT NO: 07

Title: To Understand and implement K Nearest neighbor and logistic regression for predicting values based on the given dataset

Tools: Anaconda Navigator , Jupyter Notebook

Theory:

- **K-Nearest Neighbors (KNN)** and Logistic Regression are two common classification algorithms. KNN is a non-parametric, instance-based method that classifies a new data point based on the majority class of its nearest k neighbors. It calculates distances (e.g., Euclidean) between points and is effective for small datasets but can be slow for large ones. The choice of k affects model performance—too small leads to overfitting, while too large may cause underfitting.
- **Logistic Regression** is a parametric, statistical method used for binary classification. It applies the sigmoid function to map input features to probabilities between 0 and 1. Unlike KNN, Logistic Regression learns a function during training, making it computationally efficient. However, it assumes a linear relationship between features and log-odds, limiting its effectiveness on complex, non-linear data.
- Both models require data preprocessing, such as handling missing values and scaling features. KNN benefits from feature normalization, while Logistic Regression performs well with properly transformed linear features. Evaluation metrics like **accuracy, precision, recall, and F1-score** help compare their effectiveness. The choice between them depends on dataset size, complexity, and interpretability needs.

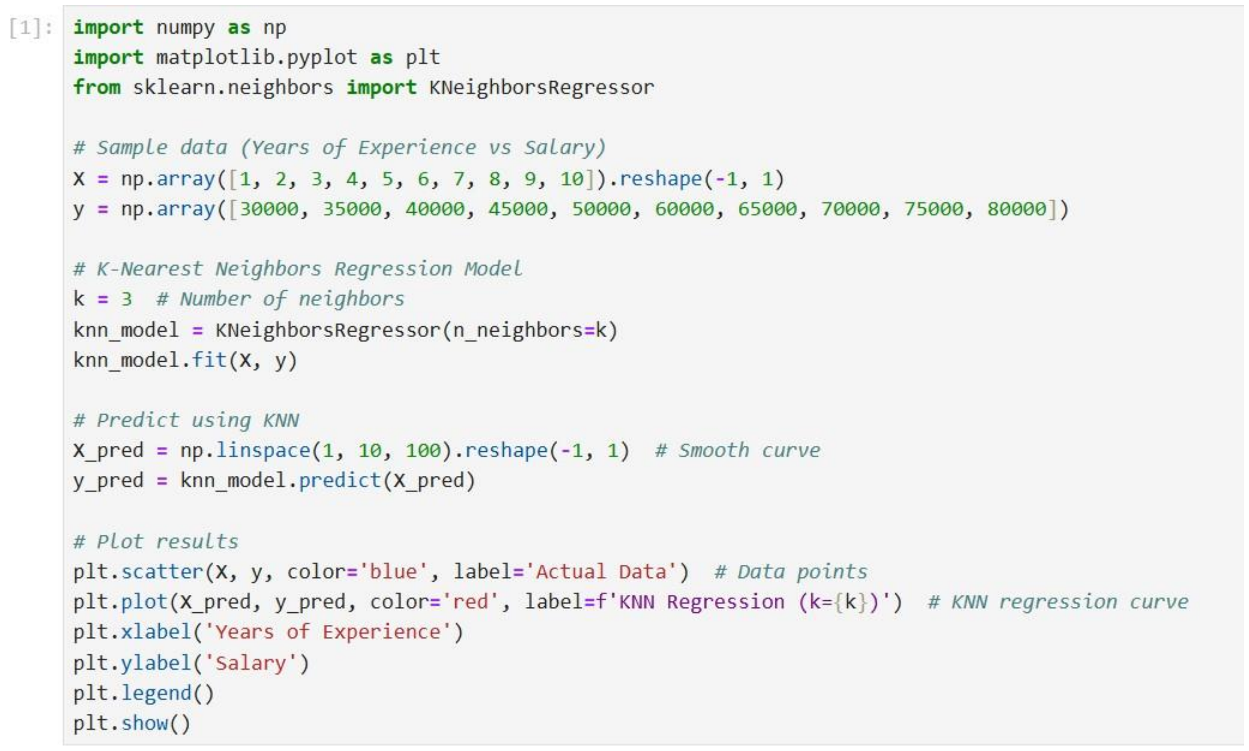
Code for K-Nearest neighbour for predicting values based on the given dataset

Code:

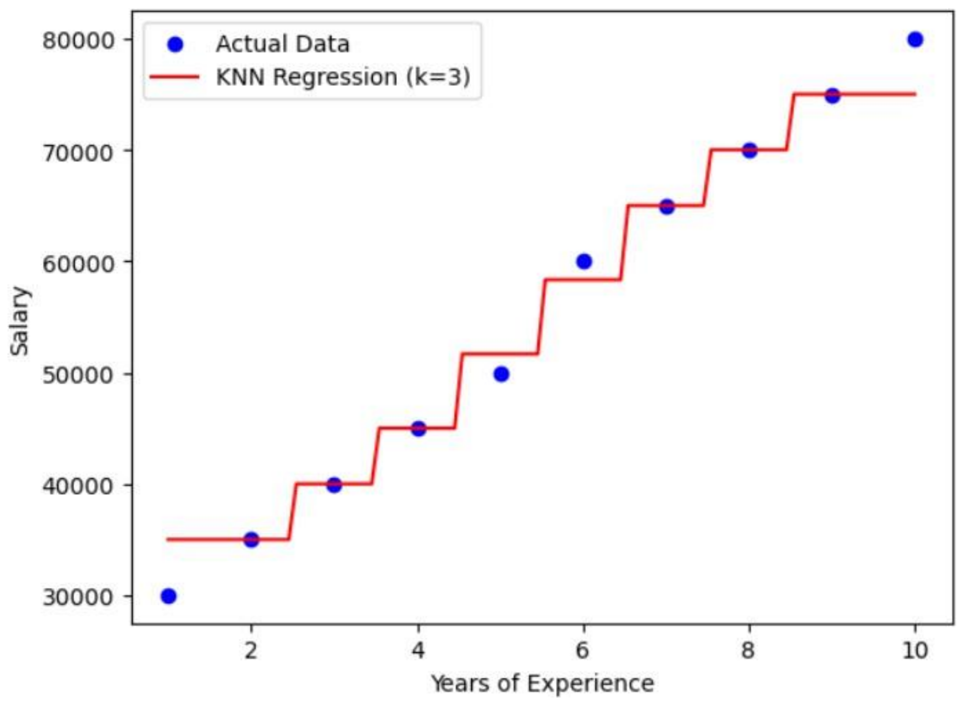
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor

# Sample data (Years of Experience vs Salary)
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)
y = np.array([30000, 35000, 40000, 45000, 50000, 60000, 65000, 70000, 75000, 80000])

# K-Nearest Neighbors Regression Model
k = 3  # Number of neighbors
knn_model = KNeighborsRegressor(n_neighbors=k)
```

```
knn_model.fit(X, y)

# Predict using KNN
X_pred = np.linspace(1, 10, 100).reshape(-1, 1)  # Smooth curve
y_pred = knn_model.predict(X_pred)

# Plot results
plt.scatter(X, y, color='blue', label='Actual Data')  # Data points
plt.plot(X_pred, y_pred, color='red', label=f'KNN Regression (k={k})')  # KNN regression curve
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.show()
```

<u>Output:</u>



```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.neighbors import KNeighborsRegressor

     # Sample data (Years of Experience vs Salary)
     X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)
     y = np.array([30000, 35000, 40000, 45000, 50000, 60000, 65000, 70000, 75000, 80000])

     # K-Nearest Neighbors Regression Model
     k = 3  # Number of neighbors
     knn_model = KNeighborsRegressor(n_neighbors=k)
     knn_model.fit(X, y)

     # Predict using KNN
     X_pred = np.linspace(1, 10, 100).reshape(-1, 1)  # Smooth curve
     y_pred = knn_model.predict(X_pred)

     # Plot results
     plt.scatter(X, y, color='blue', label='Actual Data')  # Data points
     plt.plot(X_pred, y_pred, color='red', label=f'KNN Regression (k={k})')  # KNN regression curve
     plt.xlabel('Years of Experience')
     plt.ylabel('Salary')
     plt.legend()
     plt.show()
```

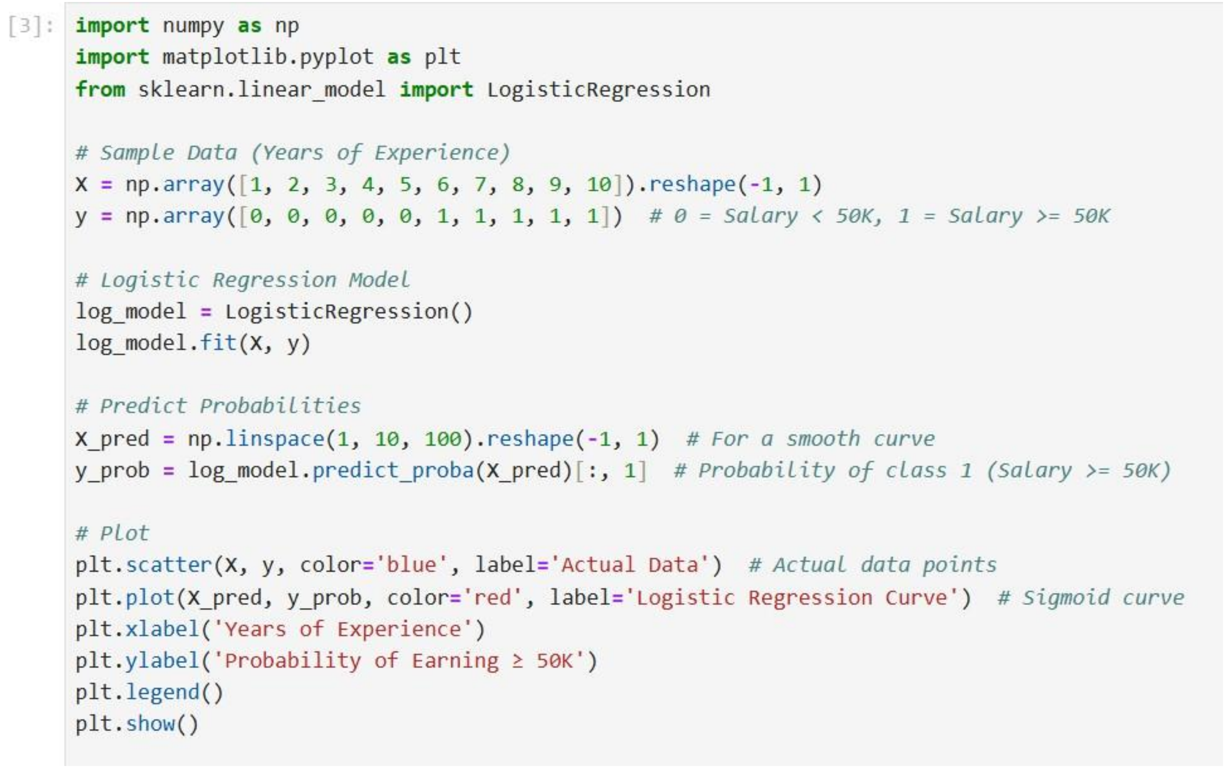


<u>Code for logistic regression for predicting values based on the given dataset</u>

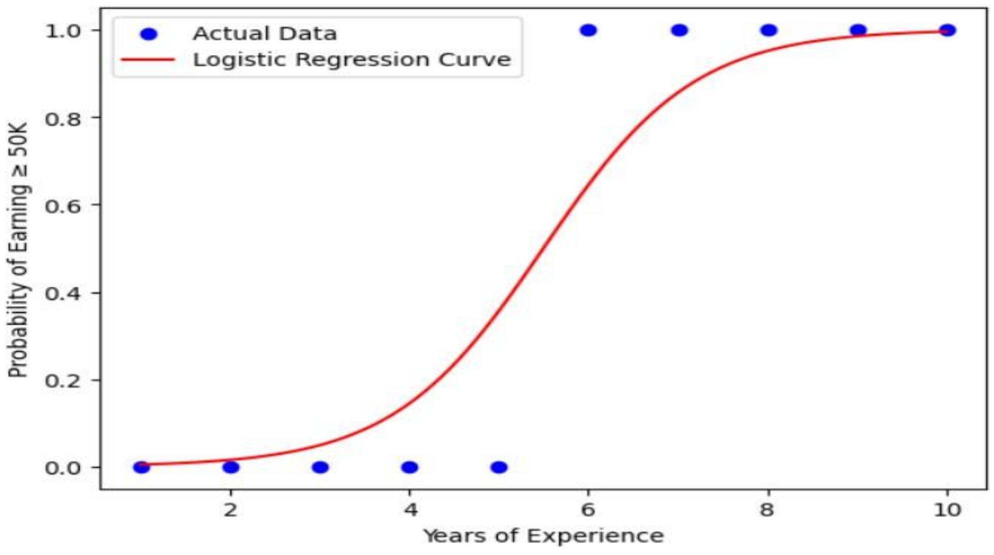
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

# Sample Data (Years of Experience)
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)
y = np.array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])  # 0 = Salary < 50K, 1 = Salary >= 50K

# Logistic Regression Model
log_model = LogisticRegression()
log_model.fit(X, y)

# Predict Probabilities
X_pred = np.linspace(1, 10, 100).reshape(-1, 1)  # For a smooth curve
y_prob = log_model.predict_proba(X_pred)[:, 1]  # Probability of class 1 (Salary >= 50K)

# Plot
plt.scatter(X, y, color='blue', label='Actual Data')  # Actual data points
plt.plot(X_pred, y_prob, color='red', label='Logistic Regression Curve')  # Sigmoid curve
```

plt.xlabel('Years of Experience')
plt.ylabel('Probability of Earning ≥ 50K')
plt.legend()
plt.show()

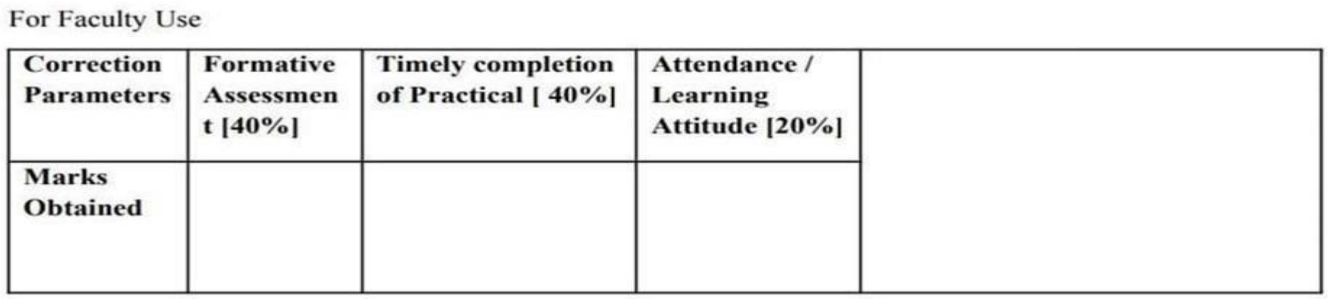
<u>Output:</u>



```
[3]: import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.linear_model import LogisticRegression

     # Sample Data (Years of Experience)
     X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)
     y = np.array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])  # 0 = Salary < 50K, 1 = Salary >= 50K

     # Logistic Regression Model
     log_model = LogisticRegression()
     log_model.fit(X, y)

     # Predict Probabilities
     X_pred = np.linspace(1, 10, 100).reshape(-1, 1)  # For a smooth curve
     y_prob = log_model.predict_proba(X_pred)[:, 1]  # Probability of class 1 (Salary >= 50K)

     # Plot
     plt.scatter(X, y, color='blue', label='Actual Data')  # Actual data points
     plt.plot(X_pred, y_prob, color='red', label='Logistic Regression Curve')  # Sigmoid curve
     plt.xlabel('Years of Experience')
     plt.ylabel('Probability of Earning ≥ 50K')
     plt.legend()
     plt.show()
```



Conclusion: In this experiment, we implemented K-Nearest Neighbors (KNN) and Logistic Regression for predicting values. KNN classifies data based on nearby points, while Logistic Regression predicts probabilities for binary classification. Both methods have their strengths—KNN is useful for non-linear patterns, while Logistic Regression is efficient for linearly separable data. The choice between them depends on the dataset, and accuracy can be improved with proper tuning and preprocessing.



For Faculty Use

Correction Parameters	Formative Assessmen t [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				