SABARISH SANKARAN B (3122215002087)

UIT2721 - Deep Learning Concepts and Architectures
Exercise 1: Classification and Regression

1. Linear Regression with Single Feature

Aim:

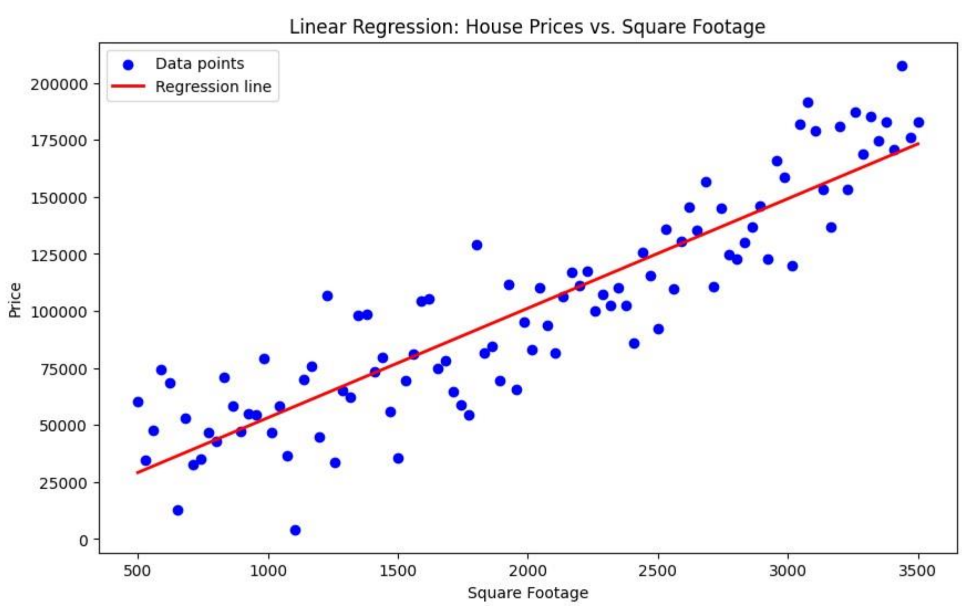
To implement linear regression on a dataset with a single feature and evaluate its performance.

Explanation:

- Linear regression predicts the target variable as a linear function of the input feature.
- It aims to find the line that best fits the data points.
- Evaluation is done using Mean Squared Error (MSE), which measures the average squared difference between predicted and actual values.

Procedure:

1. Generate or load a dataset with a single feature and target variable (e.g., house prices).
2. Split the data into training and test sets.
3. Fit a linear regression model to the training data.
4. Plot the data points and the regression line.
5. Predict the target variable for the test set.
6. Calculate and print the Mean Squared Error (MSE).



Output:

Result:

Visualized data points with the regression line and MSE value indicating model performance.

---

2. Logistic Regression for Binary Classification

Aim:

To implement logistic regression for classifying data into two classes and evaluate model performance.

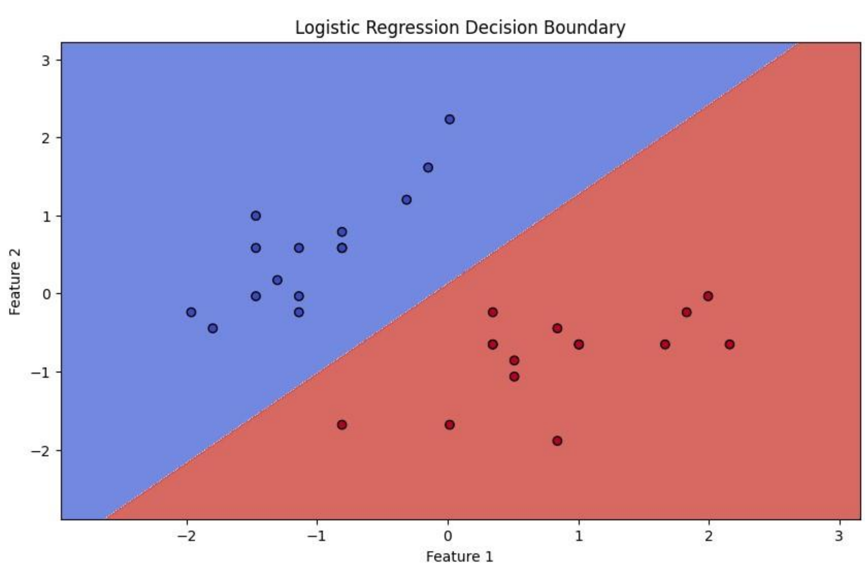
Explanation:

- Logistic regression predicts the probability of class membership using a logistic function.
- The decision boundary separates the two classes.
- Evaluation metrics include accuracy, precision, recall, and F1-score, which assess model performance from different perspectives.

Procedure:

1. Load the Iris dataset and select only two classes.
2. Split the data into training and test sets.
3. Train a logistic regression model on the training data.
4. Plot the decision boundary.
5. Evaluate the model using accuracy, precision, recall, and F1-score.

Output:



Result:

Visualized decision boundaries with evaluation metrics (accuracy, precision, recall, F1-score) indicating classification performance.

---

3. K-Nearest Neighbors (KNN) Classification

Aim:
To implement KNN classification and assess the effect of different values of kkk on accuracy.

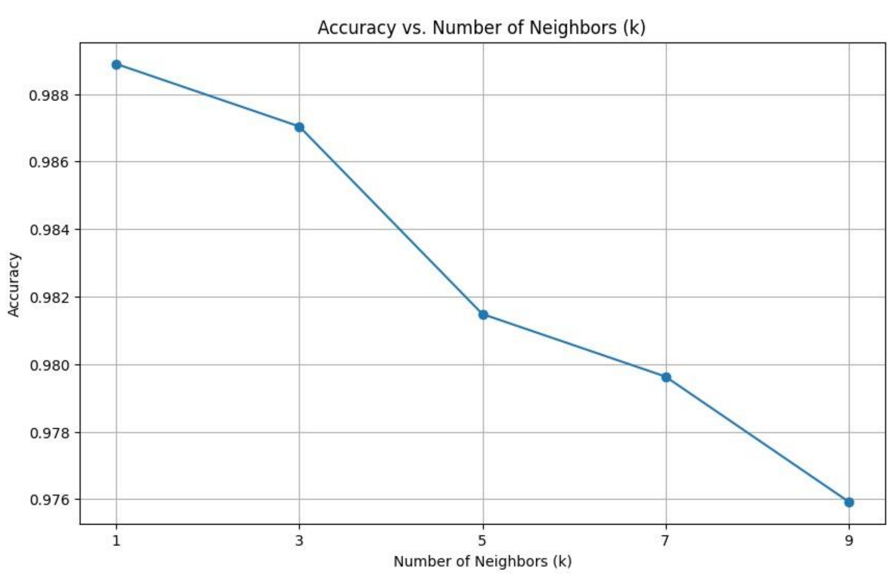
Explanation:

- KNN classifies data points based on the majority class among their kkk nearest neighbors.
- The value of kkk affects the model's sensitivity to noise and decision boundary smoothness.
- Accuracy is used to measure classification performance.

Procedure:

1. Load a dataset (e.g., handwritten digits).
2. Implement KNN classification from scratch or use scikit-learn's KNN.
3. Train and test the model with different values of kkk.
4. Evaluate and compare the accuracy for each kkk.

Output:



Result:

Observed accuracy variations with different values of kkk, showing how kkk influences model performance.

---

4. Decision Tree Classification

Aim:

To implement a decision tree classifier and analyze its structure and feature importance.

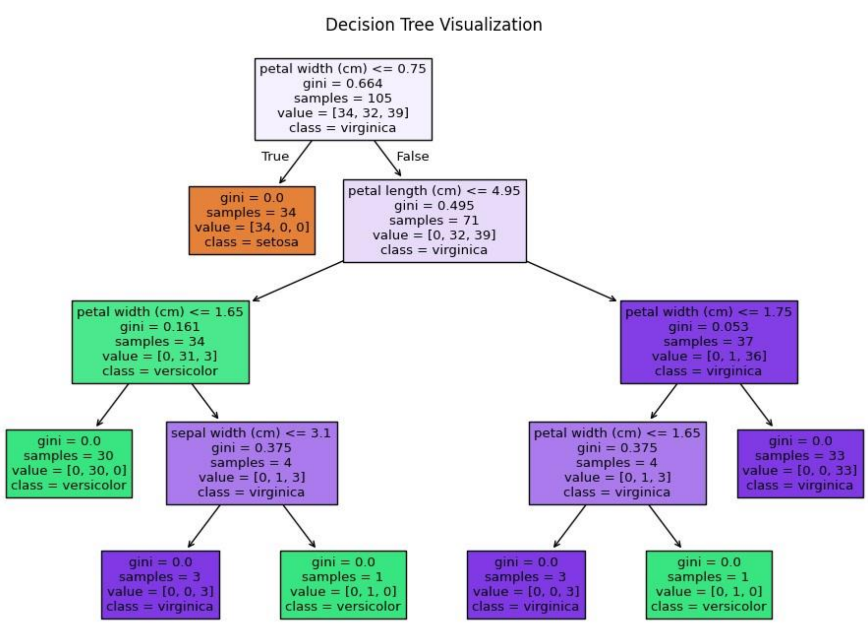
Explanation:

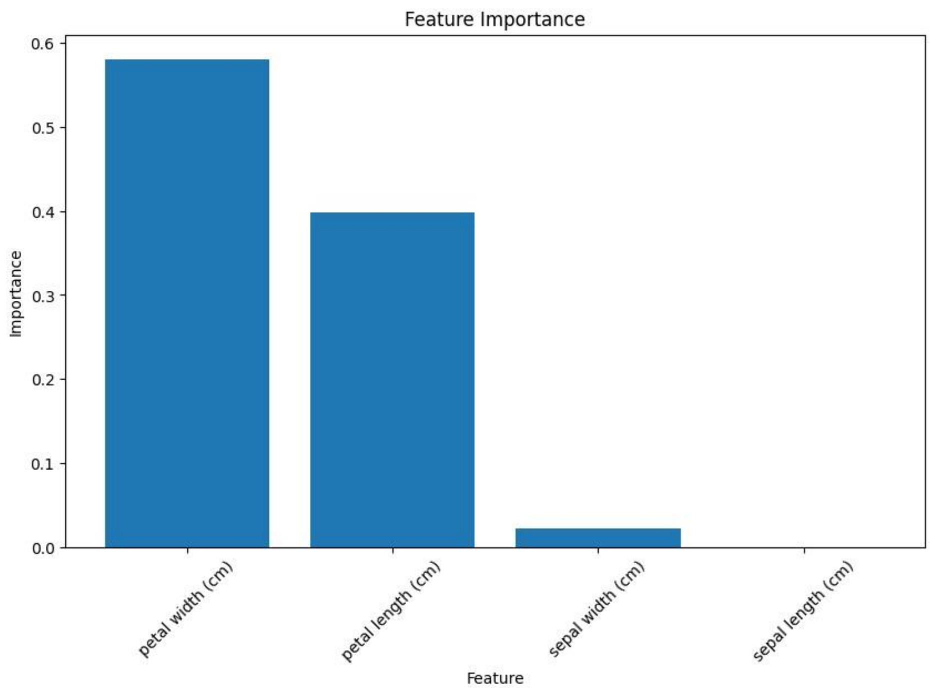
- A decision tree classifier makes decisions based on a series of rules derived from the features.
- It visualizes the decision-making process as a tree structure.
- Feature importance measures how much each feature contributes to the classification.

Procedure:

1. Load a dataset (e.g., Iris dataset).
2. Train a decision tree classifier using scikit-learn.
3. Visualize the tree structure.
4. Analyze and print the feature importance scores.

Output:





Result:
Visualized the decision tree structure and analyzed feature importance scores indicating the contribution of each feature.

---

7. Naive Bayes Classification

Aim:
To implement a Naive Bayes classifier for text classification and evaluate its performance.

Explanation:

- Naive Bayes uses Bayes' theorem with the assumption of feature independence to classify text data.
- Techniques like TF-IDF convert text into numerical features.
- Performance is evaluated using metrics such as accuracy and classification report.

Procedure:

1. Load a text classification dataset (e.g., spam detection).
2. Preprocess the text data using TF-IDF.

3. Train a Naive Bayes classifier on the preprocessed data.
4. Predict and evaluate the model using accuracy and classification report.

Output:

```
Example 1: Not Spam
Example 2: Spam
Example 3: Not Spam
Example 4: Spam
Example 5: Not Spam
Example 6: Spam
```

Result:

Evaluated the Naive Bayes classifier's performance with accuracy and classification report metrics.

---

8. Support Vector Machines (SVM) with Kernel Trick

Aim:

To implement SVM with different kernels and visualize their decision boundaries on a non-linearly separable dataset.

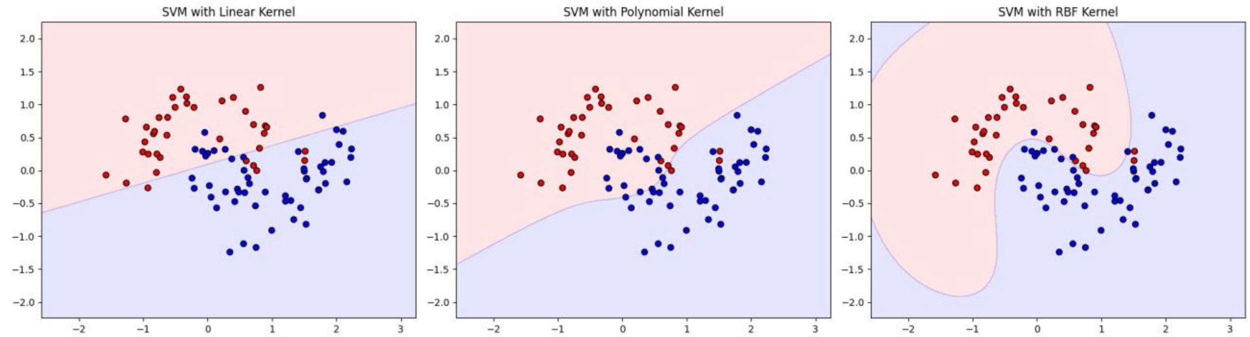
Explanation:

- SVMs find the optimal hyperplane to separate classes, with kernels transforming the data into higher dimensions to handle non-linearity.
- Common kernels include linear, polynomial, and RBF (radial basis function).
- Decision boundaries are visualized to understand how different kernels handle classification.

Procedure:

1. Generate a non-linearly separable dataset (e.g., using make_moons).
2. Train SVM classifiers with different kernels (linear, polynomial, RBF).
3. Visualize the decision boundaries for each kernel.

Output:



Result:

Visualized decision boundaries for different SVM kernels, showing how each kernel deals with non-linearly separable data.