Statistical foundation of Data Sciences

Practical- 09

Roll number: GF202344767

Name: Ishita Mehta

Workflow summary

Step 1: Exploratory Data Analysis (EDA)

EDA helps us understand the dataset before applying any model.

a) head()

Shows the first few records of the dataset.

Purpose:

- To get a quick overview of the structure
- To verify that the dataset has been loaded correctly
- To check column names & sample values

b) describe()

Provides summary statistics for each numerical feature.

- This includes:
- Mean
- Standard deviation
- Minimum & maximum values
- Quartiles (25%, 50%, 75%)

Purpose:

- Understand data distribution
- Detect extreme values
- Check if features are on different scales

c) groupby()

Grouping the dataset by species shows the mean values of features for each flower class.

Purpose:

- To see how classes differ
- To understand which features help in classification
- Identify patterns (e.g., Virginica has the largest petal length)

Step 2: Feature Scaling

- KNN is a distance-based algorithm.
- If features have different scales (e.g., petal width vs sepal length), the distance calculation becomes biased.

Why scale?

- To give all features equal importance
- To prevent large-scale features from dominating the distance metric

Method:

- Standardization (Z-score normalization)
- Each feature is transformed so that:
- Mean = 0
- Standard deviation = 1
- This makes the data suitable for KNN.

Step 3: Splitting the Dataset

Before training the model, we split the data into:

Training set (usually 80%)

Testing set (usually 20%)

Why split?

To evaluate the model on unseen data and avoid overfitting.

Step 4: Training the KNN Model

KNN classifies a new data point by looking at the K nearest neighbors (based on Euclidean distance) in the training set.

Key concept:

- The model does not learn parameters
- It stores the entire training data
- Predictions are made by “majority voting” among neighbors

Choosing initial K:

Often we start with K = 5 as it balances bias and variance.

Step 5: Making the Confusion Matrix

A confusion matrix compares actual vs predicted labels.

It shows:

- True Positives (correct classifications)
- False Positives

- False Negatives

Why useful?

- Helps identify which classes are misclassified
- Gives detailed insight beyond accuracy
- For the Iris dataset, a perfect or near-perfect matrix is common due to clean data.

Step 6: Calculating Accuracy Score

- Accuracy = (Number of correct predictions) / (Total predictions)
- Given the simplicity of the Iris dataset, KNN usually produces high accuracy, often above 95%.

Why measure accuracy?

- It provides a simple performance metric
- Useful to compare models (e.g., KNN vs Decision Tree)

Step 7: Creating the Classification Report

The classification report includes:

- Precision → How many predicted positives were correct
- Recall → How many actual positives were correctly predicted
- F1-score → Harmonic mean of precision & recall
- Support → Number of samples in each class

Why needed?

- It provides a deeper evaluation especially in multi-class problems
- Shows if any class is harder for the model to predict

Step 8: Comparing Error Rate with K Values

Since KNN depends heavily on the choice of K, we test multiple K values:

- K = 1 to 30
- For each K:
- Train the model
- Predict on the test set
- Calculate the error rate (1 − accuracy)

Why compare?

To find the value of K that gives the lowest error.

Step 9: Plotting Error vs K

A line graph is plotted with:

- X-axis → K values
- Y-axis → Error rate

Purpose:

- To visually identify the optimal K
- To avoid choosing a K that leads to overfitting (e.g., K = 1)
- To avoid underfitting with a very large K
- The best K is where the graph shows the minimum error.

Step 10: Finding the Best K

The K corresponding to the lowest error rate is selected as the optimal K.

This ensures:

- Better generalization
- Balanced bias–variance tradeoff
- Often for Iris, the best K is between 1 and 7.

Step 11: Visualizing Test Results

- A 2-feature scatter plot (usually first two features) is used:
- Each point represents a flower sample
- Colors represent predicted species

Github Repository link

https://github.com/pineapplesdontbelongonpizza/CSU1658_practical1_Testing_Pandas_and_Numpy.git