Lab - 05

KNN Algorithm

K-Nearest Neighbours (KNN) is a simple, non-parametric machine learning used for classification and regression tasks.

Step 1: Choose the number K

2) Calculate distance b/w new data point and all points in the training dataset

3) Sort the distances & select "K" nearest neighbours

4) Predict the output for
Classification and Regression

5) Return the predicted label or value

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Code (using sklearn)

```
iris = load_iris()
x = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)
y_pred = knn.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of KNN classifier:
    {accuracy:.2f}")
print(f"Predictions: {y_pred}")
print(f"True labels: {y_test}")
```

Output

Accuracy of KNN classifier: 1.00
Predictions: [1 0 2 ... 0 1 2 1 1 2 0 0 0 1 2 ...]
True labels: [1 0 2 1 0 1 2 1 1 2 0 0 0 ...]

Tuning K:

- if K is too small, the model may be noisy & overfit the data
- if K is too large, the model may be too simple & underfit the data

SVM Algorithm (Support vector machine)

a powerful algorithm used for classification, regression and outlier detection tasks.
It is a powerful and versatile model that works well both linear & non-linear data

Algorithm

Input: A dataset with labeled ex

Output: A hyperplane that best separates the classes in the feature space.

Date: / /

Training: e find the optimal hyperplane by maximizing margin btw classes.

- For non-linearly separable data, apply kernel function to transform the data into higher dimension, where a hyperplane can be found
- Prediction: The new data point is classified by determining on which side of hyperplane it lies.

Code:

```
np.random.seed(42)
n_samples = 1000
age = np.random.randint(18, 70, n_samples)
income = np.random.randint(30, 150, n_samples)
usage_freq = np.random.randint(1, 11, n_samples)
Purchase_decision = (age + income/2 + usage_freq*2 > 100).astype(int)
data = pd.DataFrame({
    'Age': age,
    'Income': income,
    'Usage_freq': usage_freq,
    'Purchase_Decision': Purchase_decision})
X = data[['Age', 'Income', 'Usage_Freq']]
y = data['Purchase_Decision']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

papergrid

Date / /

```
svm = SVC(kernel = 'rbf', c = 1, gamma = 'scale')
svm.fit(x = dtrain_scaled, y = dtrain)
y_pred = svm.predict(x = dtest_scaled)
accuracy = accuracy_score(y_test, y_pred)
```

Output

Accuracy of SVM classifier on Customer Purchase

Prediction : 0.99

Prediction : [0 1 1 1 0 0 1 0]

True Labels : 581 0

737 1

740 1

660 1

411 1

678 0

624 0

859 1

7/4