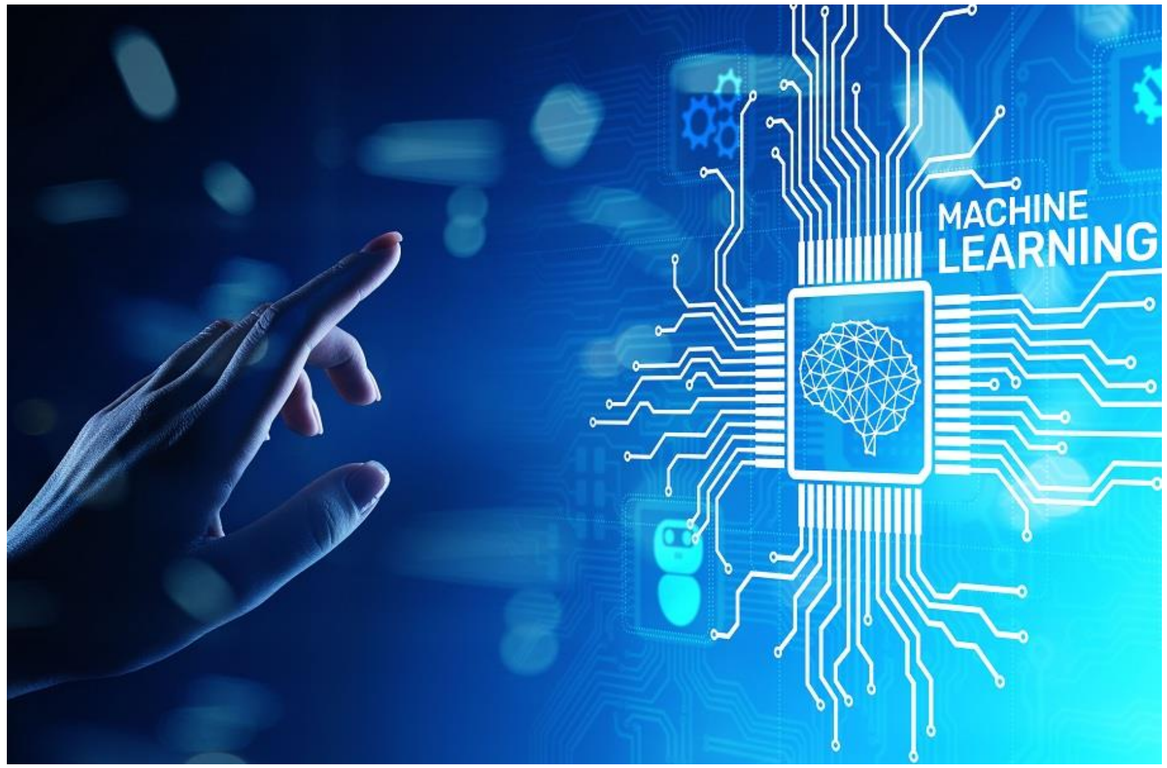
KNN, Logistic Regression



Learning for Today (03.02.2026)

- KNN
- Logistic Regression
- AUC and ROC

KNN – Sample 1

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix

/*load sepal length, sepal width, petal length, petal width in X and y
is iris.target (3 classes → 0, 1, 2*/

/* Split data in to training and testing*/

knn = KNeighborsClassifier(n_neighbors=5,
metric='euclidean')
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
```

KNN – Sample 2

- While splitting the data → stratify and check the results
- Split the data such that the class distribution in train and test remains the same as the original dataset

Sample 3 - Different k-values

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

k_values = range(1, 11)
accuracy = []


for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy.append(accuracy_score(y_test, y_pred))

plt.plot(k_values, accuracy, marker='o')
plt.xlabel("K value")
plt.ylabel("Accuracy")
plt.title("Accuracy vs K")
plt.show()
```

Sample 4 – Feature Scaling

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

print("Accuracy after scaling:",
      accuracy_score(y_test, knn.predict(X_test)))
```

Sample 5 – knn regression

```
from sklearn.neighbors import KNeighborsRegressor


knn_reg = KNeighborsRegressor(n_neighbors=2)
knn_reg.fit(X, y)

print("Predicted value for 3.5:",
      knn_reg.predict([[3.5]]))
```

Exercises

1. Apply the KNN (Scaled features) for the below 2 datasets

    - https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/code
    - https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/code

Logistic Regression

Sample 6: Logistic Regression on Dataset from kaggle

https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/code

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report

data = load_breast_cancer()

X = # Features
y = # Labels (0 = malignant, 1 = benign)
```

```
/* Split dataset to train and test */

/* Perform Feature Scaling */


model = LogisticRegression()
model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)
y_prob = model.predict_proba(X_test_scaled)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n",
confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n",
classification_report(y_test, y_pred))
```

Sample 7: Sigmoid probability Visualization

```
/*Create X_Feature by selecting one feature from X */

/*Split train and test data */

/*Feature Scaling*/

/*Create logisticregression model, fit with train data*/


x_values = np.linspace(X_train.min(), X_train.max(),
100).reshape(-1, 1)
y_values = model.predict_proba(x_values)[:, 1]

plt.plot(x_values, y_values)
plt.xlabel("Feature value")
plt.ylabel("Probability")
plt.title("Sigmoid Curve")
plt.show()
```

Sample 8: ROC and AUC

ROC plots:

- **X-axis:** False Positive Rate (FPR)
- **Y-axis:** True Positive Rate (TPR)

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

AUC = Area Under the ROC Curve

AUC = 1 → perfect classifier

AUC = 0.5 → random guessing

To add the previous code, add the below code

```
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc_score = roc_auc_score(y_test, y_prob)

rint("AUC Score:", auc_score)

 lt.plot(fpr, tpr, label=f"AUC = {auc_score:.2f}")
plt.plot([0, 1], [0, 1], linestyle='--')  # Random classi
line
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Logistic Regression")
plt.legend()
plt.show()
```

Exercise

2. Apply the logistic regression (Scaled features) for the below 2 datasets (with ROC & AUC)

- https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/code
- https://www.kaggle.com/datasets/uciml/iris/code