BAKU HIGHER OIL SCHOOL

INFORMATION TECHNOLOGY DEPARTMENT
INFORMATION SECURITY
DIVISION

Machine Learning

Home Assignment 3

Assignment name: KNN Algorithm

Student name: Huseyn Abdullayev

Group number: CS 25

Instructor: Leyla Muradkhanli

MASTER OF CYBER SECURITY BAKU 2025

Dataset Description and Preprocessing Steps

The dataset used for this assignment is the Titanic Survival Dataset. It contains 891 samples of passengers from the Titanic shipwreck, with information on whether they survived or not.

The dataset includes the following key columns:

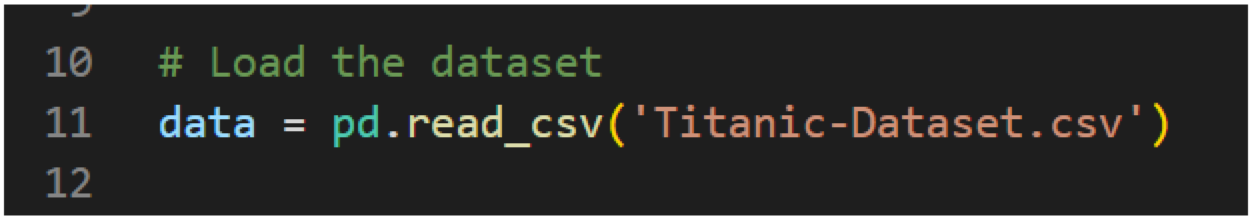
- PassengerId: Unique identifier for each passenger.
- Survived: Target variable (0 = Did not survive, 1 = Survived).
- Pclass: Ticket class (1 = 1st, 2 = 2nd, 3 = 3rd).
- Name: Passenger name.
- Sex: Gender (male or female).
- Age: Age in years.
- SibSp: Number of siblings/spouses aboard.
- Parch: Number of parents/children aboard.
- Ticket: Ticket number.
- Fare: Passenger fare.
- Cabin: Cabin number.
- Embarked: Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton).

The target variable is "Survived," which is binary (categorical).

Data Preprocessing Steps:

Loading Data:

The dataset was imported using pandas.read_csv('Titanic-Dataset.csv').



```python
# Load the dataset
data = pd.read_csv('Titanic-Dataset.csv')

```

Data Cleaning:

Missing values were handled as follows:

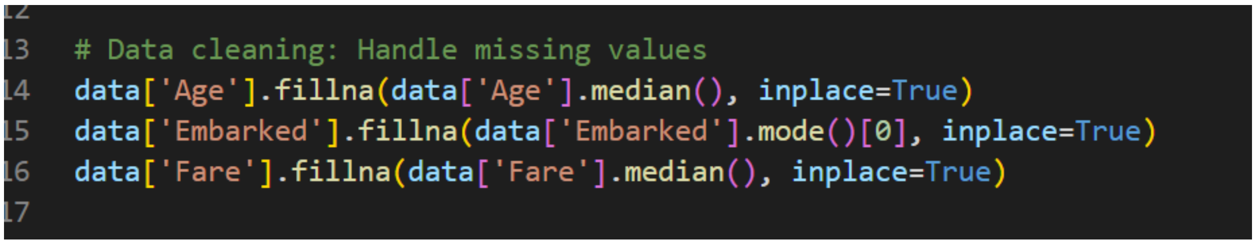
Age: Filled with median value (177 missing).

Embarked: Filled with mode value (2 missing).

Fare: Filled with median value (0 missing, but checked).

Cabin: Ignored due to high missing values (687 missing) and irrelevance.

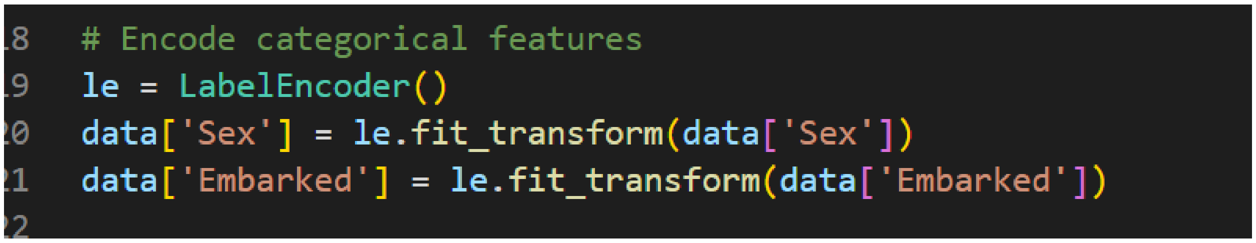
No other columns had missing values that required handling.



```python
# Data cleaning: Handle missing values
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
data['Fare'].fillna(data['Fare'].median(), inplace=True)
```

Encoding Categorical Labels:

Categorical features "Sex" (male/female) and "Embarked" (C/Q/S) were encoded into numeric labels using LabelEncoder from sklearn.preprocessing.



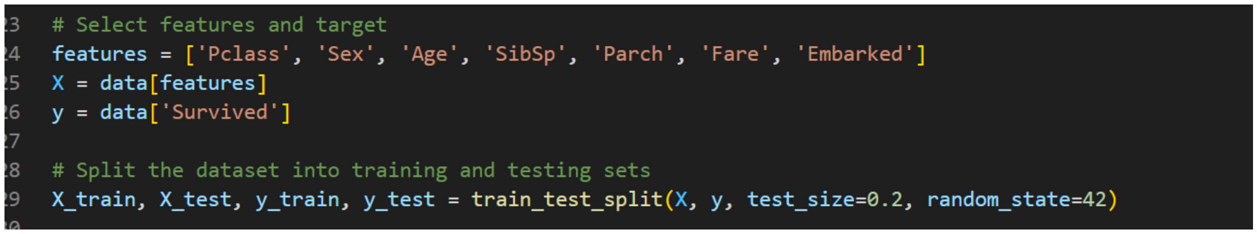
```python
# Encode categorical features
le = LabelEncoder()
data['Sex'] = le.fit_transform(data['Sex'])
data['Embarked'] = le.fit_transform(data['Embarked'])
```

Feature Selection and Splitting:

Selected features: Pclass, Sex, Age, SibSp, Parch, Fare, Embarked (as X).

Target: Survived (as y).

The dataset was then split into 80% training data (712 samples) and 20% testing data (179 samples) using train_test_split() with random_state=42 for reproducibility.

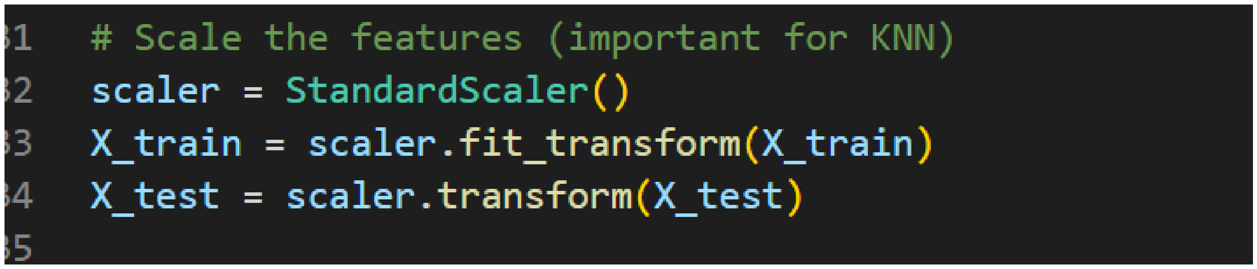


```python
# Select features and target
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
X = data[features]
y = data['Survived']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

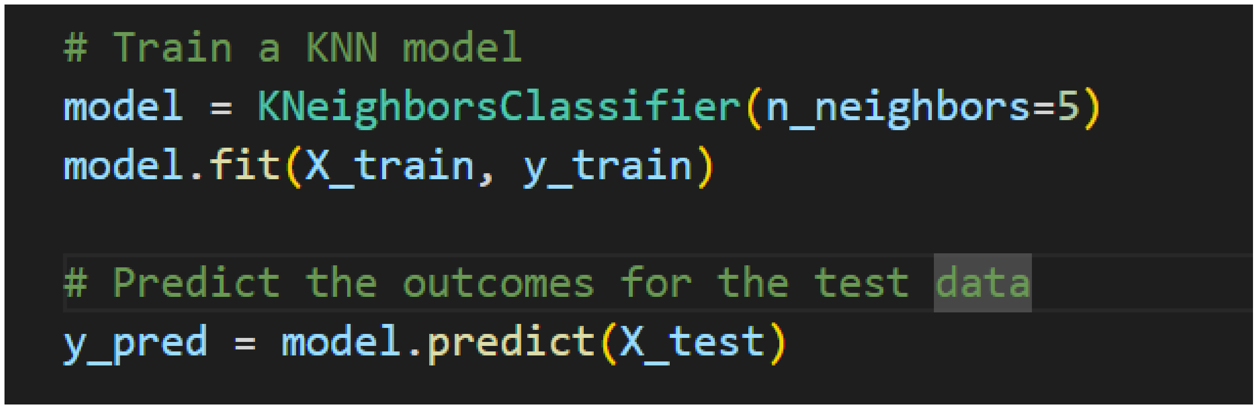
Summary of Model Implementation

Since KNN is distance-based, features were scaled using StandardScaler to standardize them.



```python
# Scale the features (important for KNN)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

A K-Nearest Neighbors (KNN) model from sklearn.neighbors was chosen for this binary classification task. The model was trained using the training data (712 samples) with n_neighbors=5.

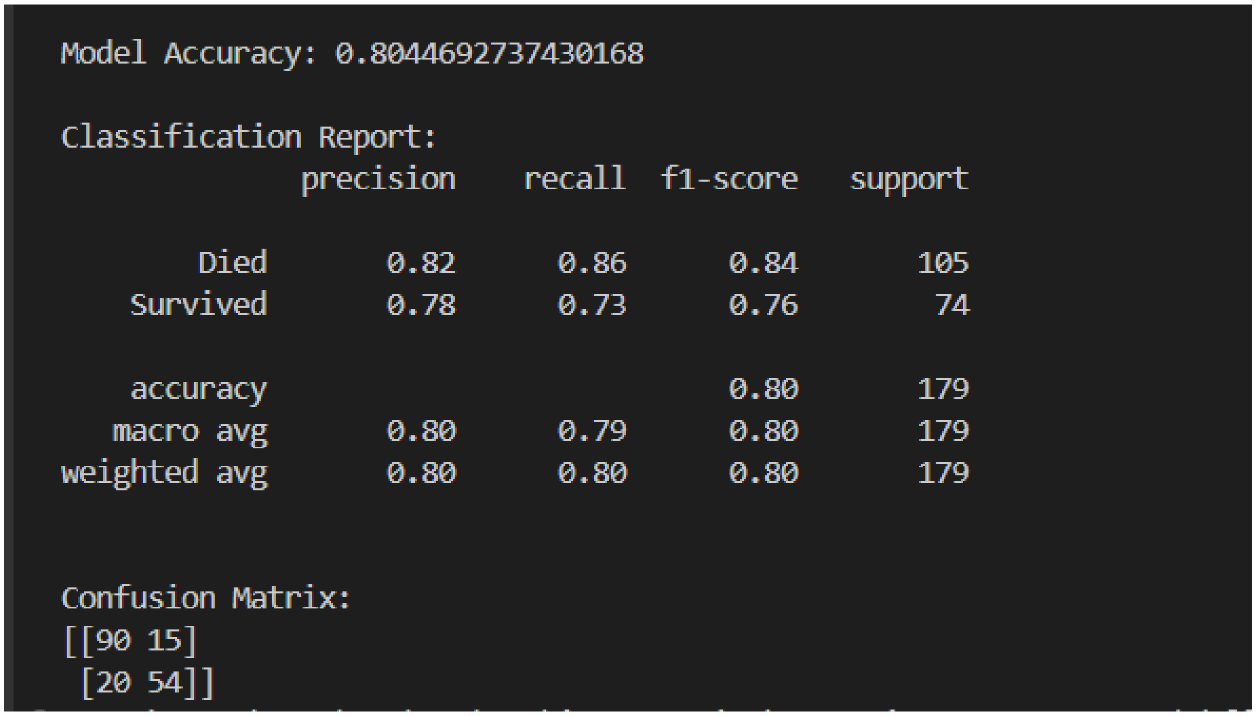


```python
# Train a KNN model
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)

# Predict the outcomes for the test data
y_pred = model.predict(X_test)
```

KNN does not have coefficients like logistic regression; it classifies based on the majority vote of the 5 nearest neighbors in the feature space.

Predictions were made on the test data, and performance metrics were calculated.



```
Model Accuracy: 0.8044692737430168

Classification Report:
              precision    recall  f1-score   support

        Died       0.82      0.86      0.84       105
    Survived       0.78      0.73      0.76        74

    accuracy                           0.80       179
   macro avg       0.80      0.79      0.80       179
weighted avg       0.80      0.80      0.80       179


Confusion Matrix:
[[90 15]
 [20 54]]
```

Interpretation and Conclusion

The model performs reasonably, with stronger prediction for non-survivors, similar to historical patterns where fewer people survived. Sex and Pclass likely influence neighbors due to their strong correlation with survival. The KNN classifier achieved 80% accuracy on the Titanic dataset. This demonstrates that KNN is suitable for classification tasks where data points with similar features tend to have similar outcomes. While slightly lower than some parametric models like logistic regression, it is non-parametric and can capture non-linear relationships. In real-world applications, KNN's simplicity is an advantage, but computation can be slow for large datasets. Further improvements could include tuning k or using feature engineering. This experiment shows KNN's effectiveness with structured data.