KNN-Cheat Sheet

Algorithm Overview:

- KNN is a supervised learning algorithm used for classification and regression tasks.
- It classifies a new data point based on the similarity to its neighbouring data points.
- KNN is a non-parametric algorithm, which means it does not make any assumptions about the underlying data distribution.

Steps in KNN:

Step 1: Choose the number of neighbours (K) to consider.
Step 2: Calculate the distance between the new data point and all existing data points.
Step 3: Select the K nearest neighbours based on the calculated distances.
Step 4: For classification, count the occurrences of each class among the K neighbours and assign the new data point to the majority class.
Step 5: For regression, calculate the average or weighted average of the target values of the K neighbours and assign it to the new data point.

Scikit-Learn Implementation:
Step 1: Import the necessary modules:

```
from sklearn.neighbors import
KNeighborsClassifier, KNeighborsRegressor
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
StandardScaler
from sklearn.metrics import accuracy_score,
mean_squared_error
```

Step 2: Split the data 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)
```

Step 3: Preprocess the data (e.g., scaling):

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

Step 4: Create and train the KNN classifier or regressor:

```
clf = KNeighborsClassifier(n_neighbors=5)
clf.fit(X_train_scaled, y_train) reg =
KNeighborsRegressor(n_neighbors=5)
reg.fit(X_train_scaled, y_train)
```

Step 5: Make predictions:

```
y_pred_cls = clf.predict(X_test_scaled)
y_pred_reg = reg.predict(X_test_scaled)
```

Step 6: Evaluate the model:

```
accuracy = accuracy_score(y_test,
y_pred_cls) mse =
mean_squared_error(y_test, y_pred_reg)
```

Advantages of KNN:

- Simple to implement and understand.
- Robust to noisy training data.
- Effective when the training data is large.

Disadvantages of KNN:

- Need to determine the value of K, which can be complex.
- High computational cost as it calculates distances for all training samples.
- Performance can be sensitive to the choice of distance metric.

Use Cases:

- Recommendation systems.
- Credit card fraud detection.
- Handwriting recognition (OCR).

Saira Sanadi TYCSE C64