KNN (K nearest neighbours)

MUDES DATE: 24/12/2023

- KNN (K nearest neighbours) is one of the simplest algorithms we will learn about!
- Section Overview:
  - KNN Theory and Intuition
  - KNN Classification Coding Example
  - KNN exercise Overview
  - KNN Exercise Solution:

- While KNN can be used for regression tasks, its performance can be quite poor and less efficient than other algorithm, so we've decided not to exibit its use for regression.
- However if you do want to use it for regression it is very easy to swap in the KNN Regressor model with scikit-learn

- K mean algorithm is different then KNN

ISLR Relevant Reading:-
- chapter 2

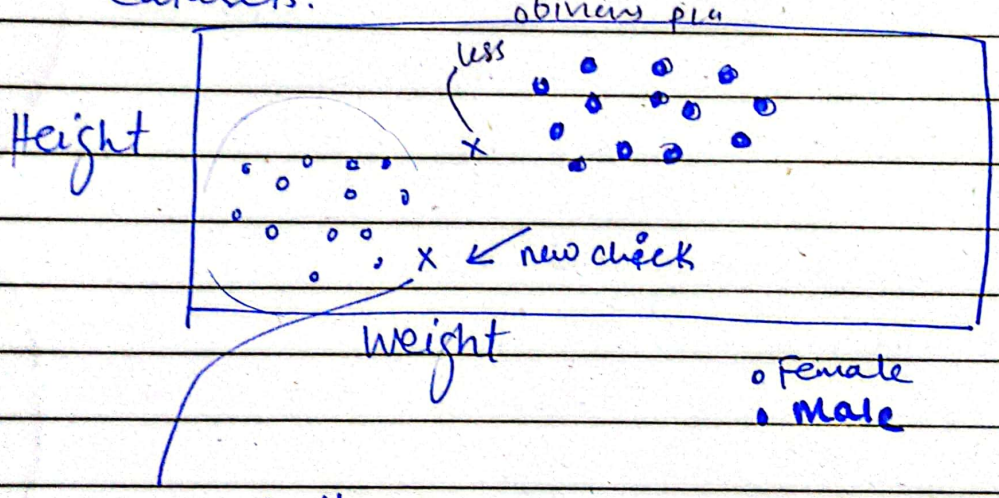
M T W T F S DATE:

2- KNN Classification Theory and Intuition

- K nearest neighbours is one of the simplest machine learning algorithms.
- Its simply assign a label to new data based on the distance between the old data and new data.
- lets go through the intuition with an example use case.

- **Sexing chicks is still a very manual process:**

- Imagine a height and weight datasets.

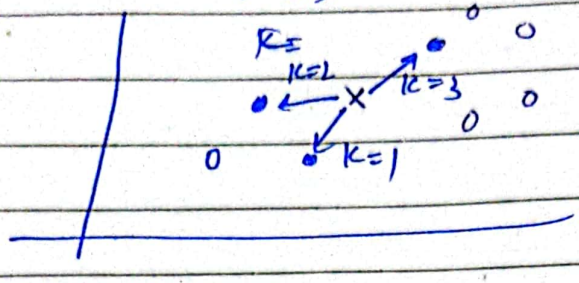


- We intuitively "know" this is likely female - Intuition comes from distance to points.

- What about ~~less~~ obious points, how many points should be consider

NOTES DATE:

let Image Sigution like this



when K = 4, 2 there could be a posibily tie how to chose when this condition occurs.

Tie consideration and options:

- Always chose an odd tie.
- Inease of tie, simply reduce K by 1 untill tie is broken.
- Randomly break tie
- Choose the nearest class

What does scikit-learn do in case of tie?

- Warning: Regarding the Nearest Neighbours algorithms, if it is found that two neighbour, neighbours K+1 and K ~~are~~ have identical distances but different labels, the result will depend on the order of the training data.

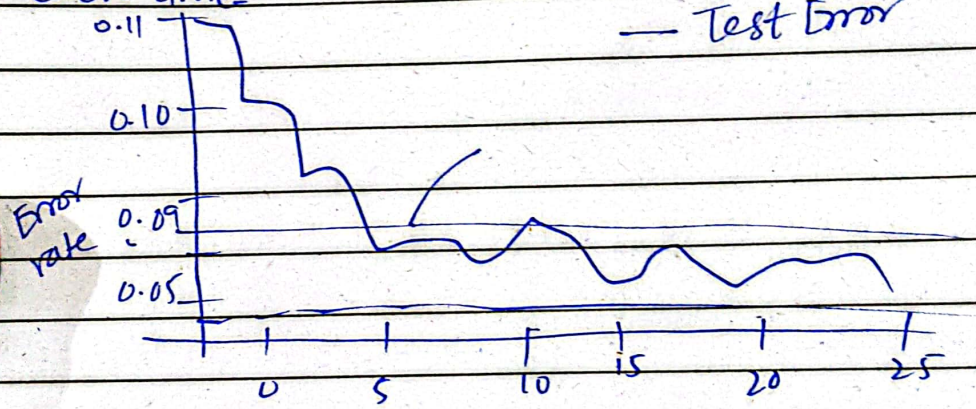
- In case of tie, the answer will be the class that happens to appear first in the sets of neighbour.
- Results are ordered by distance, so it choose the class of the closet point.

M T W T F S DATE:

- We want K value that minimizes error: Error = 1 - Accuracy
- Two methods:
  - Elbows methods.
  - Cross validation a grid search of multiple K value and chose K that result in lowest error or highest accuracy.

- Elbow Method:-

Train your model on multiple K-values and plot the error ~~on test~~ after testing each time.



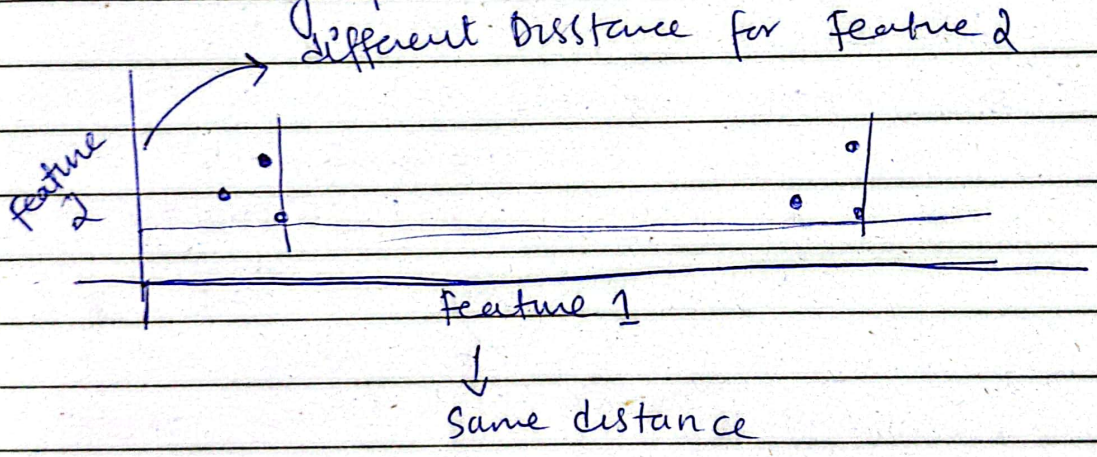
- Cross validation only takes into account the K value with the lowest error rate across multiple folds.
- This could result in a more complex model (higher value of K).
- Consider the context of problem to decide if K values are an issue.

MUDIFS DATE:_____

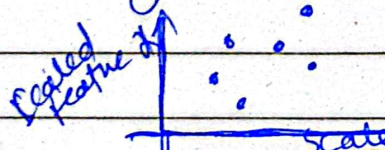
- KNN Algorithm
  - Choose K values
  - Sort features vectors (N-dimensional space) by a distance metric.
  - Choose class based on K nearest feature vectors.

- KNN consideration:-
  - Distance Metrics
  - Many ways to measure distance:
    - Minkowski
    - Euclidean
    - Manhattan
    - Chebyshev

- Scaling for Distance



- ~~Scaling for Dis~~ feature could have vastly different Ranges!
- Scaling is a necessary value for KNN.



M T W T F S DATE:

- While KNN algorithm is relatively simple, keep in mind the following considerations:
  - choosing the optimal K value.
  - Scaling features.
- Lets continue to explore how to explore KNN for classification-

03. KNN Classification

Coding part 1: Data and Model

```
# Coding
Import numpy as np
import pandas as pd

df= pd.read_csv("gene_expression")
df.head()
sns.scatterplot(data=df, x='Gene One',
                y="gene two",
                hue="cancer points",
                alpha=0.8,
                style="cancer present")
plt.xlim(2,4)
plt.ylim(4,7)

sns.pairplot(data=df, hue="patient cancer")
```

MIRIES DATE:

```
from sklearn.model_selection import train
from ll .preprocessing ll StandardScaler

X = df.drop("patient Cancer", axis=1)
y = df["patient cancer"]

scater ftes X-t, X.test, y-t, yt = train_test_
                                   split(--)

Scaler = StandardScaler()
Scaled_X_train = Scaler.fit_transform(X_train)
Scaled_X_test = Scaler.transform(X_test)

from sklearn.neighbours import KNeighborsClassifier
help(n_neighbors)

knn_model = KNeighborsClassfier(n_neighbors=1)
knn_model.fit(Scaled_X_train, y_train)
y_preds = knn_model.predict(Scaled_X_test)

from sklearn.metrics import confusion_
                            matrix,
                            classification_report
confusion_matrix(y_test, y_preds)

print(classification_report(ytest, ypreds)

df['cancer present'].value_counts()
```

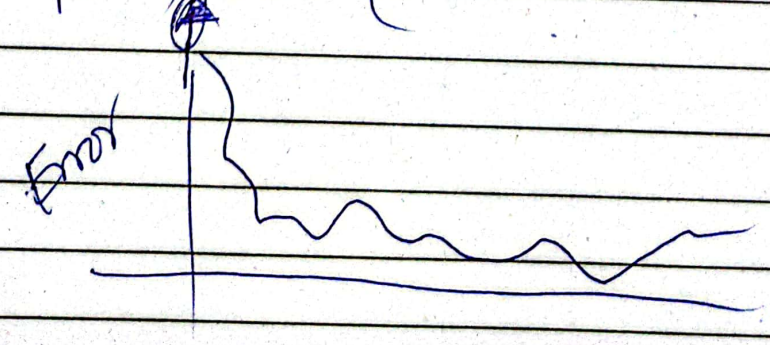
M T W T F S DATE:

04 KNN Classification:
Coding part Two: choosing K

- A pipeline object in scikit-learn can set up a sequence of repeated operation, such as a scaler and a model.
- This way only the pipeline needs to be called, instead of having to repeatly call a scaler and a model.

```
# 1
test_error_rates = [ ];
for k in range(1,30):
    knn_model = KNeighborsClassifier(n_neighbors=k)
    KNN_model.fit(scaled_X_train, y_train)
    ypreds_test = knn_model.predict(scaled_X_test)
    test_error = 1 - accuracy_score(y_test, ypreds_test)
    test_errors.append(test_error)

plt.plot(range(1,30), test_error_rates)
plt.xlabel(" ")
plt.ylabel(" ")
```



MIDTERMS DATE:

Setting up pipeline

scaler = StandardScaler()
knn = KNeighborsClassifier()
knn.get_params().keys()

operations = [('scaler', scaler), ('knn', knn)]

from sklearn.pipeline import Pipeline

pipe = Pipeline(operations)

from sklearn.model_selection import GridSearchCV

k_value = list(range(1,20))
param_grid = {'knn__n_neighbors': k_values,

full_cv_classifier = GridSearchCV(pipe, param_grid, cv=5, scoring='accuracy')

full_cv_classfier.fit(X_train, y_train)

full_cv_classifier.best_estimator_.get_params()

full_cv_classifier.predict(X_test)

full_preds = full_.