1/11/22 Friday

Naive Bayes

Topic :- **Naive Bayes** — Naive Bayes classifier is a machine learning algorithm based on Supervised learning, that can be used for Solving classification problem.

It is a probabilistic classifier, which means it predict on the basis of the probability of an object.

i) Comprises of two Nouns Naive Bayes !

- Naive — It is called Naive because it makes the assumption that all attributes are independent of each other
- Bayes — It's called bayes because it depends on the principle of 'Bay's theorem.

ii) where Naive Bayes is used ? —

- Face Recognition software
- weather forecast
- News categorization (Eg - political news, fitness news, Film news, Sport news)
- Real-time prediction

(iii) Independent Intution of Naive Bayes

$\Rightarrow P_r(A \cap B) = P(A) * P_r(B/A)$

$= P_r(B \cap A) = P_r(B \cap A)$

$= P_r(A) * P(B/A) = P(B) * P_r(A/B)$

$P(B/A) = \frac{P_r(B) * P_r(A/B)}{P(A)}$ ⇒ Naive Baye

⇒ Baye's theorem

$P_r(B/A)$ = B given A = conditional Probability

(iv) Types of Naive Bayes classifier —

- Multinomial Naive Bayes
- Bernoulli Naive Bayes
- Gaussian Naive Bayes — Normal Distribution

Code of Naive Bayes —

Code no ① >>> import numpy as np
>>> import pandas as pd.

Code no ② >>> from sklearn.datasets import load_breast_cancer
>>> data = load_breast_cancer()

Code no ③ >>> data.data
>>> data.feature_names
>>> data.target
>>> data.target_names

Code no ④ ∴ Our dataset is in array So we have to change it into DataFrame as we directly import it from sklearn.

>>> df = pd.DataFrame(np.c_[data.data, data.target],
columns = [list(data.feature_names)+['target']])

Code no ⑤ >>> x = df.iloc[:, 0:-1] (# independence Variable)
>>> y = df.iloc[:, -1] (# dependent Variable)

Code no ⑥ >>> from sklearn.model_selection import train_test_split
>>> x_train, x_test, y_train, y_test = train_test_split(
x, y, test_size = 0.2, random_
state = 2020)

import naive_bayes & train the data —

Code (7)
```
>>> from sklearn.naive_bayes import GaussianNB
>>> classifier = GaussianNB()
>>> classifier.fit(x_train, y_train)
```

Code (8)
```
>>> classifier.score(x_test, y_test)
>>> classifier.predict(x_test)
>>> y_test
```
(to compare)

Note :- we can also classify & check other naive-bayes classifier like MultinomialNB or BernoulliNB

Naive Bayes

$P(A|B) = \frac{P(A|B) \times P(B)}{P(A)}$ → for Categorical dataset

$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$ → for Categorical dataset

KNN

1. **KNN** → K-Nearest Neighbor is Supervised ML algorithm that can perform both classification and regression tasks using number (K) of neighbors (Instance.

KNN - Classifier

- K value should be odd.

- Steps of prediction of New instance -
  - Get Data.
  - Define K Neighbors.
  - Calculate the Neighbors distance.
  - Assign New instances to Majority of Neighbors.

To calculate the distance -

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

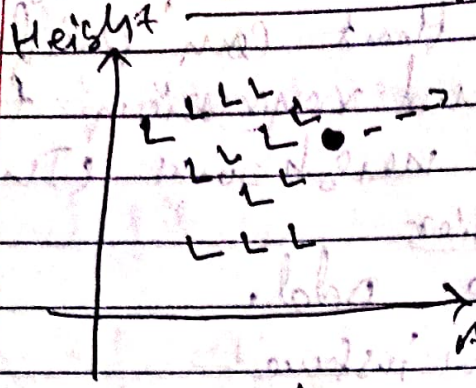
Manhattan dist = $d = |x_2 - x_1| + |y_2 - y_1|$

|| → Mode Mode

```
>>> from sklearn.neighbors import KNeighborsClassifier
>>> classifier = KNeighborsClassifier(n_neighbors = 5)
>>> classifier.fit(x_train, y_train)
```

Note If there is so much Outlier in data set than KNN will not work

KNN Regression



If K = 5 then we will find average of 5 point near the predict value.

```
>>> from sklearn.neighbours import KNeighborRegressor
>>> regression = KNeighborsRegressor(n_neighbors=5)
>>> regression.fit(x_train, y_train)
```

In KNN Regression the output will be the Average of the (K=5) & K nearest value.

Limitation of KNN

- we can't use in large dataset
- Outlier & Scaling
- Sensitive to missing value

Parameter in KNN

Weight parameter in KNN :- If distance is less weight may be High if distance is more then weight will be less

# P = take a distance = 1 = for manhattan distance
2 = for euclidean distance

∴ We can use Hyper parameter tuning after we KNN with Grid Search CV & Random Search CV.