**Logistic Regression**

**Cancer use case**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

​

# ML library

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

#1. Collecting the data

data = pd.read\_csv('./Datasets/data.csv')

data

​#2. Data Wrangling

data.drop('Unnamed: 32', axis=1, inplace=True)

data.head()

# id is irrelevant to the machine learning algorithm

data.drop('id', axis=1, inplace=True)

data.head()

import seaborn as sns

sns.countplot(data['diagnosis'], label='Count')

plt.show()

# data['diagnosis'] = data['diagnosis'].map({'M':1, 'B':0})

# data.head()

##3. Analyze the Data

train, test = train\_test\_split(data, test\_size=0.3)

print(train.shape)

print(test.shape)

train.head()

test.head()

# relevant columns

data.head()

prediction\_var = ['radius\_mean', 'texture\_mean', 'perimeter\_mean', 'area\_mean',

'smoothness\_mean','compactness\_mean','concavity\_mean', 'concave points\_mean','symmetry\_mean','fractal\_dimension\_mean']

# input training data

train\_X = train[prediction\_var]

# output training data

train\_Y = train['diagnosis']

# same steps for testing data

# input testing data

test\_X = test[prediction\_var]

# outut testing data

test\_Y = test['diagnosis']

#4. Training the model

logistic\_model = LogisticRegression()

logistic\_model.fit(train\_X, train\_Y)

#5. Testing the algorithm

predicted\_value = logistic\_model.predict(test\_X)

# accuracy

metrics.accuracy\_score(test\_Y, predicted\_value)

pd.DataFrame({'predicted\_value': predicted\_value, 'Known O/P':test\_Y})

Random Forest

ensemble of many decision trees

from sklearn.ensemble import RandomForestClassifier

# number of decision tree is 100

model = RandomForestClassifier(n\_estimators=100)

​

#training

model.fit(train\_X, train\_Y)

​

# testing

prediction = model.predict(test\_X)

​

# accuracy

metrics.accuracy\_score(test\_Y, prediction)

**Linear Regression**

**Boston House price Prediction**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#1. Data Collection

boston\_data = pd.read\_csv('BostonHousing.csv')

boston\_data.head()

#2. Data Wrangling

# input or independent variables

X = boston\_data.iloc[:,0:13]

# output or dependent variable

Y = boston\_data['medv']

print(X)

print(Y)

#3. Data Analysis

# correlation

import seaborn as sns

# create a correlation matrix

correlation = boston\_data.corr()

correlation

sns.heatmap(correlation)

plt.yticks(rotation=0)

plt.xticks(rotation=90)

plt.show()

#4. Training the algorithm

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,

Y,

test\_size=0.30,

random\_state=10)

X\_train.head()

X\_test.head()

#Y\_train.head()

from sklearn.linear\_model import LinearRegression

# model instantiation

lin\_model = LinearRegression()

# fir the model with the input and corresponding output data

lin\_model.fit(X\_train, Y\_train)

#5. Test algorithm

predicted\_house\_price = lin\_model.predict(X\_test)

plt.scatter(Y\_test, predicted\_house\_price)

plt.show()

predicted\_house\_price

pd.DataFrame({"predicted price": predicted\_house\_price,

'actual price': Y\_test}).head()

X.head()

lin\_model.coef\_

lin\_model.intercept\_

**naive\_bayes\_classifier\_weather**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix

**Load the data**

data = pd.read\_csv('pacific.csv')

data.head()

**data wrangling**

pre\_columns = data[:]

pre\_columns.drop(['ID','Name','Date','Time','Event','Status','Latitude','Longitude'],

axis=1, inplace=True)

pre\_columns.head()

relevant\_cols = pre\_columns.columns

relevant\_cols

## split the data in training and testing

train, test = train\_test\_split(data, test\_size=0.3, random\_state=5)

train\_X = train[relevant\_cols]

train\_Y = train['Status']

test\_X = test[relevant\_cols]

test\_Y = test['Status']

train\_X.shape

test\_X.shape## lets understand the output

import seaborn as sns

sns.countplot(data['Status'])

plt.show()

# use the Model

gnb = GaussianNB()

# train the model

gnb.fit(train\_X, train\_Y)

# prediction

predicted\_gnb = gnb.predict(test\_X)

# testing the predicted values

accuracy\_score(test\_Y, predicted\_gnb)

**Confusion Matrix**

Refer the Notes.doc file to understand Confusion matrix

cnf\_matrix = confusion\_matrix(test\_Y, predicted\_gnb)

cnf\_matrix

From Wiki: