**INDEX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S. No** | **Name of Experiment** | **Date of Submission** | **Date of Evaluation** | **Total Marks** | **Marks Obtained** | **Remarks/ Signature** |
| 1 | **Python Environment** | 17.01.2022 | 07.03.2022 |  |  |  |
| 2 | **Pandas Commands** | 24.01.2022 | 07.03.2022 |  |  |  |
| 3 | **Linear Regression** | 31.01.2022 | 07.03.2022 |  |  |  |
| 4 | **Logistic Regression** | 07.02.2022 | 07.03.2022 |  |  |  |
| 5 | **Gradient Descent** | 14.02.2022 | 07.03.2022 |  |  |  |
| 6 | **Missing Values** | 28.02.2022 | 07.03.2022 |  |  |  |
| 7 | **Underfitting & Overfitting** | 07.03.2022 | 11.04.2022 |  |  |  |
| 8 | **K-means Cluster** | 14.03.2022 | 11.04.2022 |  |  |  |
| 9 | **SVM for Cancer classification** | 28.03.2022 | 11.04.2022 |  |  |  |
| 10 | **Open-Ended** | 04.04.2022 | 11.04.2022 |  |  |  |

**EXPERIMENT-7**

**AIM: To implement overfitting and underfitting on a dataset.**

**THEORY:**

**Underfitting:**   
A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data. Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have fewer data to build an accurate model and also when we try to build a linear model with fewer non-linear data.

**Techniques to reduce underfitting:**

1. Increase model complexity
2. Increase the number of features, performing feature engineering

**Overfitting:**   
A statistical model is said to be overfitted when we train it with a lot of data *(just like fitting ourselves in oversized pants!)*. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too many details and noise.

**Techniques to reduce overfitting:**

1. Increase training data.
2. Reduce model complexity.
3. Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).

**CODE**:

from sklearn.datasets import make\_classification

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

from sklearn.metrics import accuracy\_score

from sklearn.neighbors import KNeighborsClassifier

from matplotlib import pyplot

# create dataset

X, y = make\_classification(n\_samples=10000, n\_features=20, n\_informative=5,

n\_redundant=15, random\_state=1)

# split into train test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)

# define lists to collect scores

train\_scores, test\_scores = list(), list()

# define the tree depths to evaluate

values = [i for i in range(1, 51)]

# evaluate a decision tree for each depth

for i in values:

# configure the model

model = KNeighborsClassifier(n\_neighbors=i)

# fit model on the training dataset

model.fit(X\_train, y\_train)

# evaluate on the train dataset

train\_yhat = model.predict(X\_train)

22

train\_acc = accuracy\_score(y\_train, train\_yhat)

train\_scores.append(train\_acc)

# evaluate on the test dataset

test\_yhat = model.predict(X\_test)

test\_acc = accuracy\_score(y\_test, test\_yhat)

test\_scores.append(test\_acc)

# summarize progress

print('>%d, train: %.3f, test: %.3f' % (i, train\_acc, test\_acc))

# plot of train and test scores vs number of neighbors

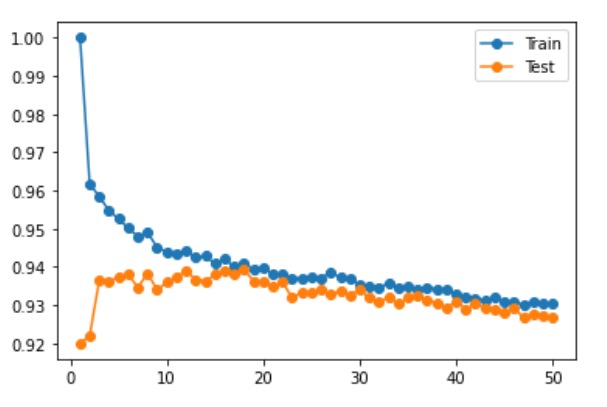
pyplot.plot(values, train\_scores, '-o', label='Train')

pyplot.plot(values, test\_scores, '-o', label='Test')

pyplot.legend()

pyplot.show()

**OUTPUT:**



**EXPERIMENT- 8**

**AIM:** To use any data to apply K-Means Algorithms

**DATASET USED:** Synthetic Data created with the help of numpy and pandasin the form a numpy array

**THEORY:**

As discussed we have created the synthetic data and used it to train our K-eans Clustering Model.

1. Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. In this algorithm we determine 2 things:

//.

* Determines the best value for K center points or centroids by an iterative process.
* Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

**CODE & OUTPUT:**

import matplotlib.pyplot as plt

%matplotlib inline

import numpy as np

from sklearn.cluster import KMeans

X = np.array([[5,3],

     [10,15],

     [15,12],

     [24,10],

     [30,45],

     [85,70],

     [71,80],

     [60,78],

     [55,52],

     [80,91],])

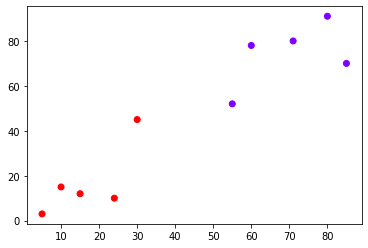
kmeans = KMeans(n\_clusters=2)

kmeans.fit(X)

print(kmeans.cluster\_centers\_)

print(kmeans.labels\_)

plt.scatter(X[:,0],X[:,1], c=kmeans.labels\_, cmap='rainbow')

****

**EXPERIMENT- 9**

**AIM:** To build a SVM model for Cancer classification

**DATASET USED:** Scikit learn breast caner dataset

**THEORY:**

A Support Vector Machine (SVM) is a binary linear classification whose decision boundary is explicitly constructed to minimize generalization error. It is a very powerful and versatile Machine Learning model, capable of performing linear or nonlinear classification, regression and even outlier detection.

SVM is well suited for classification of complex but small or medium sized datasets.

**CODE & OUTPUT:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.svm import SVC

%matplotlib inline

#Import Cancer data from the Sklearn library

# Dataset can also be found here (http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29)

from sklearn.datasets import load\_breast\_cancer

cancer = load\_breast\_cancer()

df\_cancer = pd.DataFrame(np.c\_[cancer['data'], cancer['target']], columns = np.append(cancer['feature\_names'], ['target']))

df\_cancer.head()

sns.pairplot(df\_cancer, vars = ['mean radius', 'mean texture', 'mean perimeter', 'mean area',

'mean smoothness'] )

df\_cancer['target'].value\_counts()

plt.figure(figsize=(20,12))

sns.heatmap(df\_cancer.corr(), annot=True)

X = df\_cancer.drop(['target'], axis = 1)

y = df\_cancer['target']

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 = 20)

svc\_model = SVC()

svc\_model.fit(X\_train, y\_train)

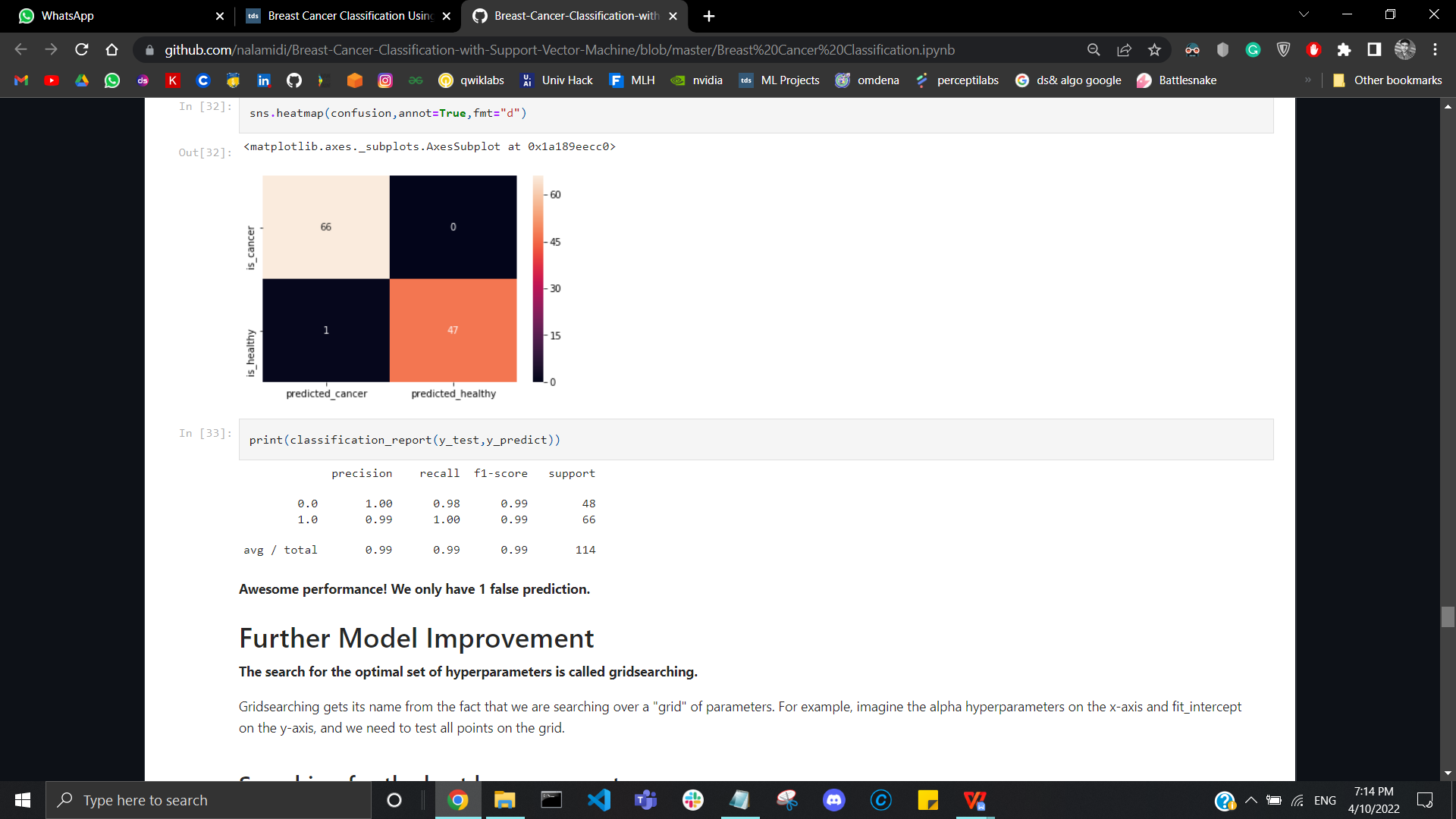
y\_predict = svc\_model.predict(X\_test)

cm = np.array(confusion\_matrix(y\_test, y\_predict, labels=[1,0]))

confusion = pd.DataFrame(cm, index=['is\_cancer', 'is\_healthy'],

columns=['predicted\_cancer','predicted\_healthy'])

print(classification\_report(y\_test, y\_predict))



**EXPERIMENT- 10**

**AIM:** To build a model for classification and regression using random forest algorithm

**DATASET USED:**

**THEORY:**

**CODE & OUTPUT:**