

JOSEPH KINGORI NJIRI

TASK 5 HACKATHON

BREAST CANCER PROGNOSIS USING MACHINE LEARNING SOLUTIONS

Each data has a unique ID as an attribute, the prognosis will give out whether a patient has cancer or not using M(malignant) or B(benign). The data will classify this outcome using the attributes the radius of the infection/lobes, the texture, compactness, area and the perimeter of tumor. The link of the dataset is <https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset>. I will fine tune the hyper parameters and evaluation metrics of various classifications.

Problem Statement

Breast cancer is the most common cancer amongst women in the world. It accounts for 25% of all cancer cases, and affected over 2.1 Million people in 2015 alone. It starts when cells in the breast begin to grow out of control. These cells usually form tumors that can be seen via X-ray or felt as lumps in the breast area. The key challenges against its detection is how to classify tumors into malignant (cancerous) or benign (non-cancerous). I will use machine learning for analysis and classifying these tumors. Tests such as MRI, mammogram, ultrasound and biopsy are commonly used to diagnose breast cancer performed.

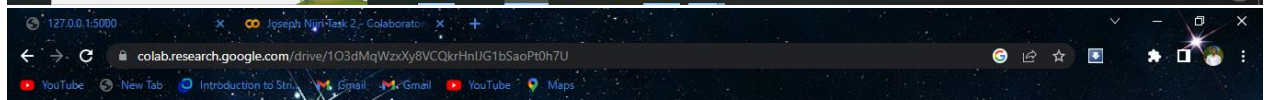
Objectives

1. Identifying the problem and Data Sources
2. Exploratory Data Analysis
3. Pre-Processing the Data
4. Build model to predict whether breast cell tissue is malignant or Benign

Exploratory Data Analysis

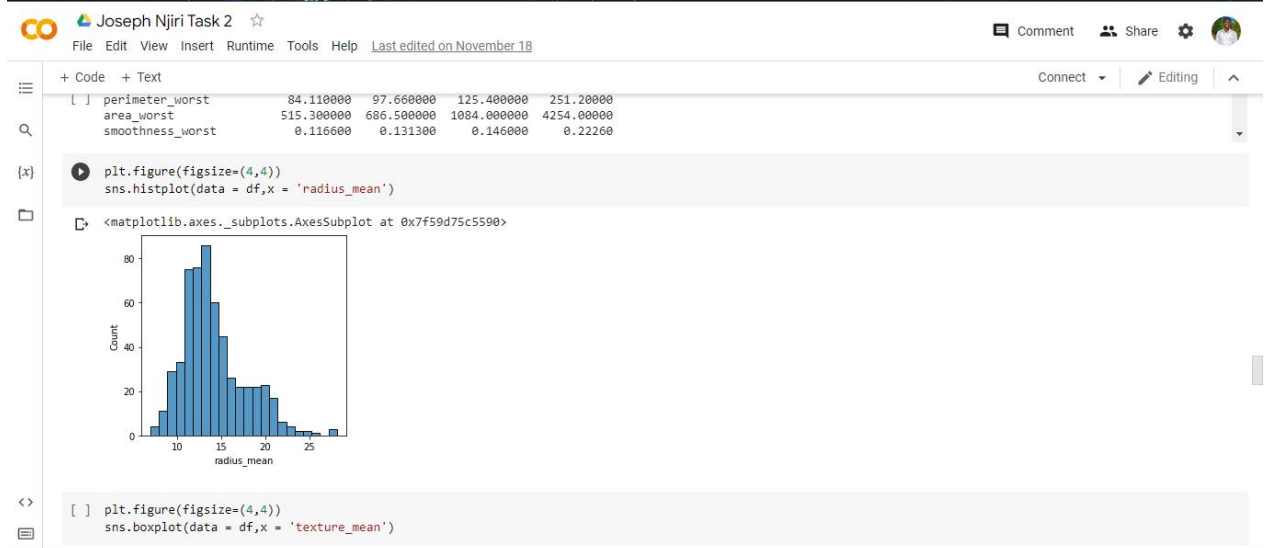
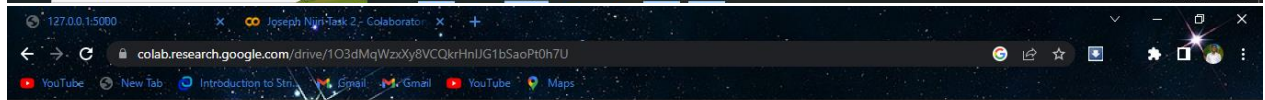


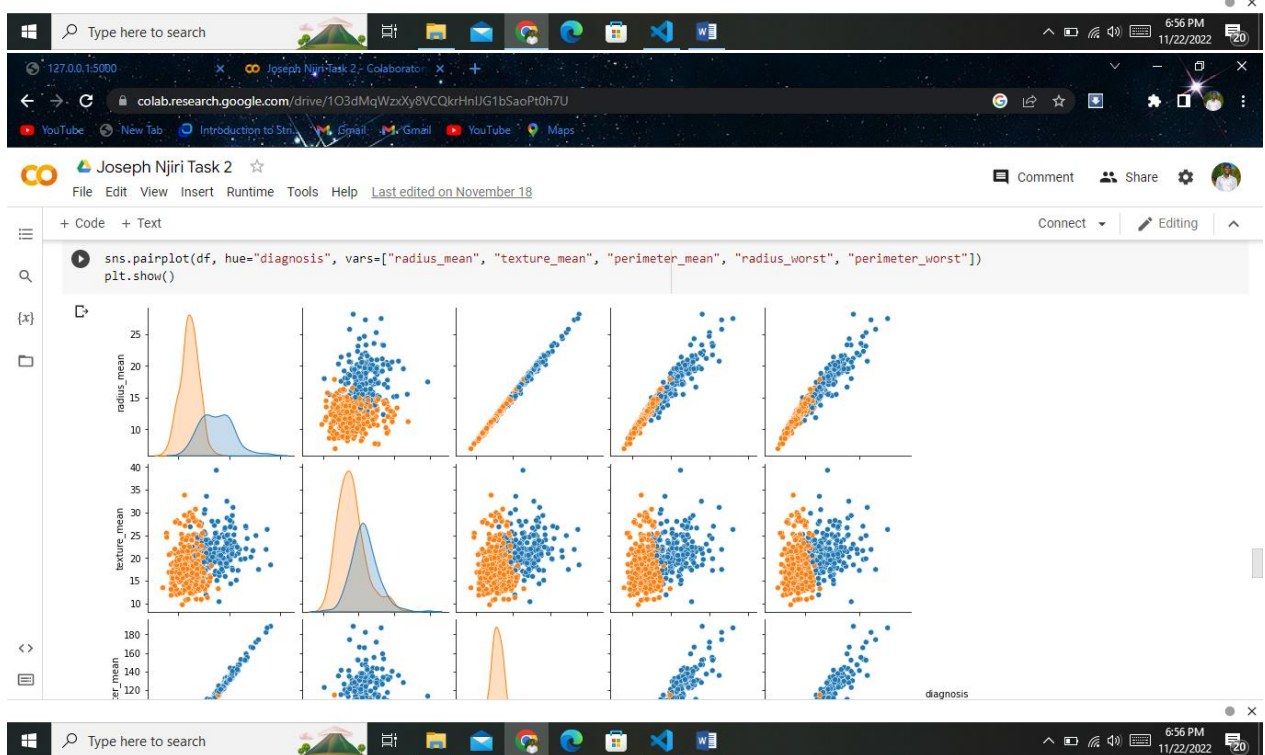
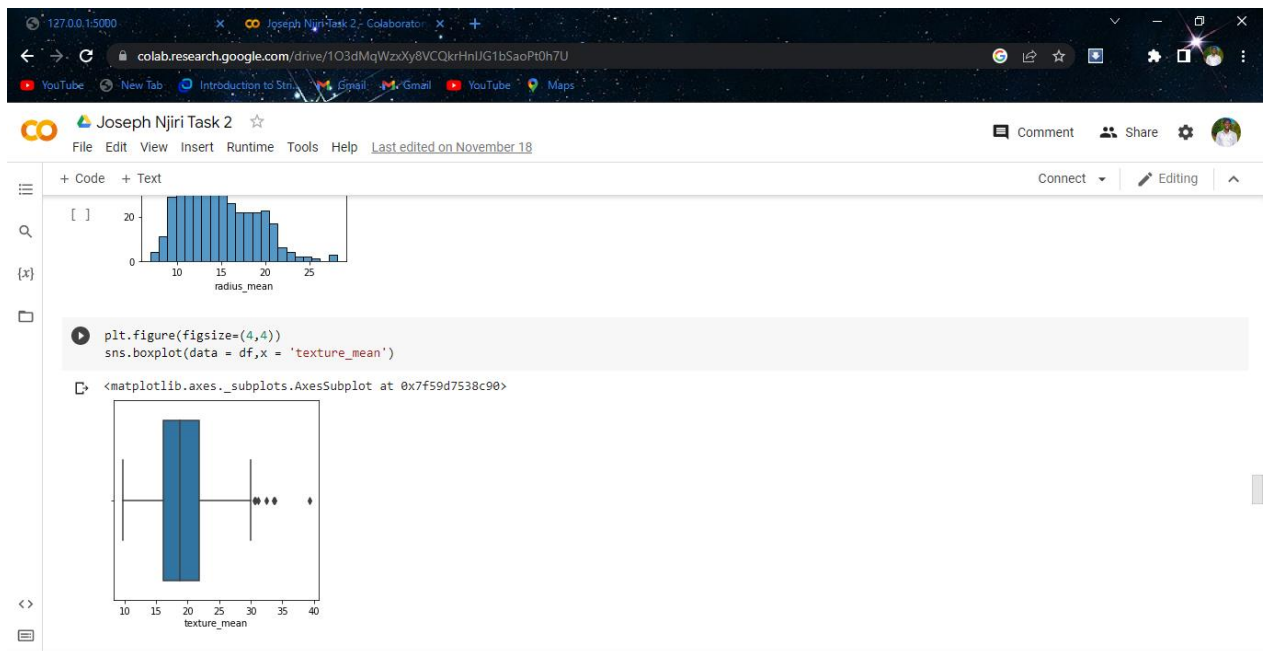
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	radius_worst	texture_worst
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	25.38	16.99
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	24.99	15.85
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	23.57	15.91
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	14.91	15.42
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	22.54	16.02
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	15.47	16.04
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.07400	22.88	16.02

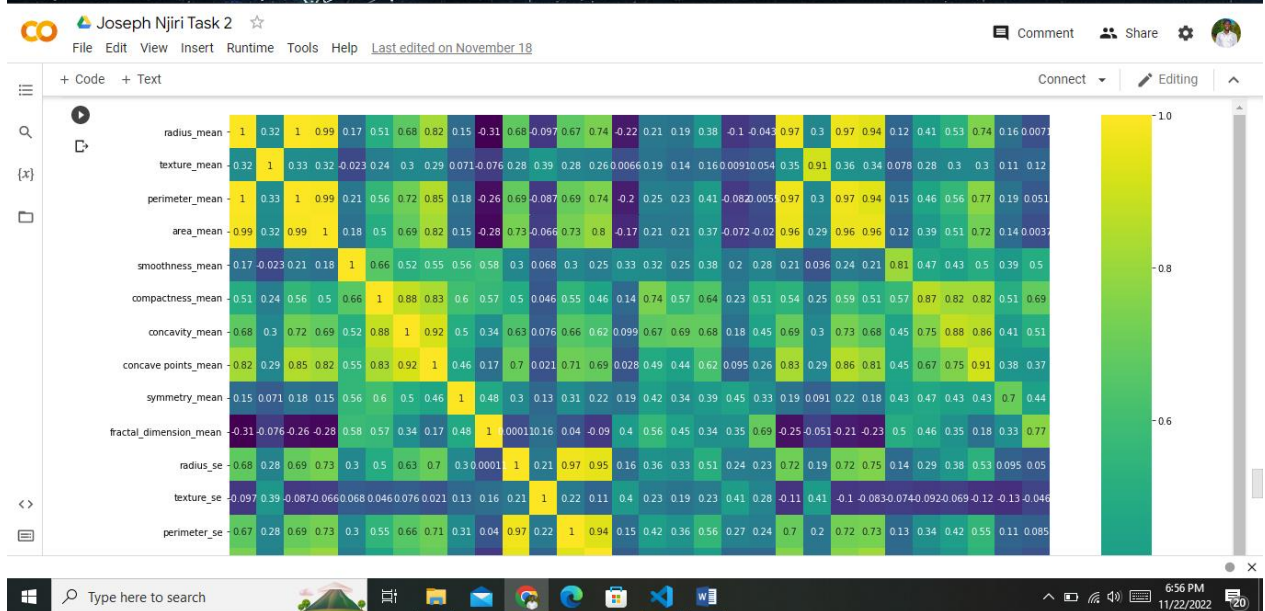
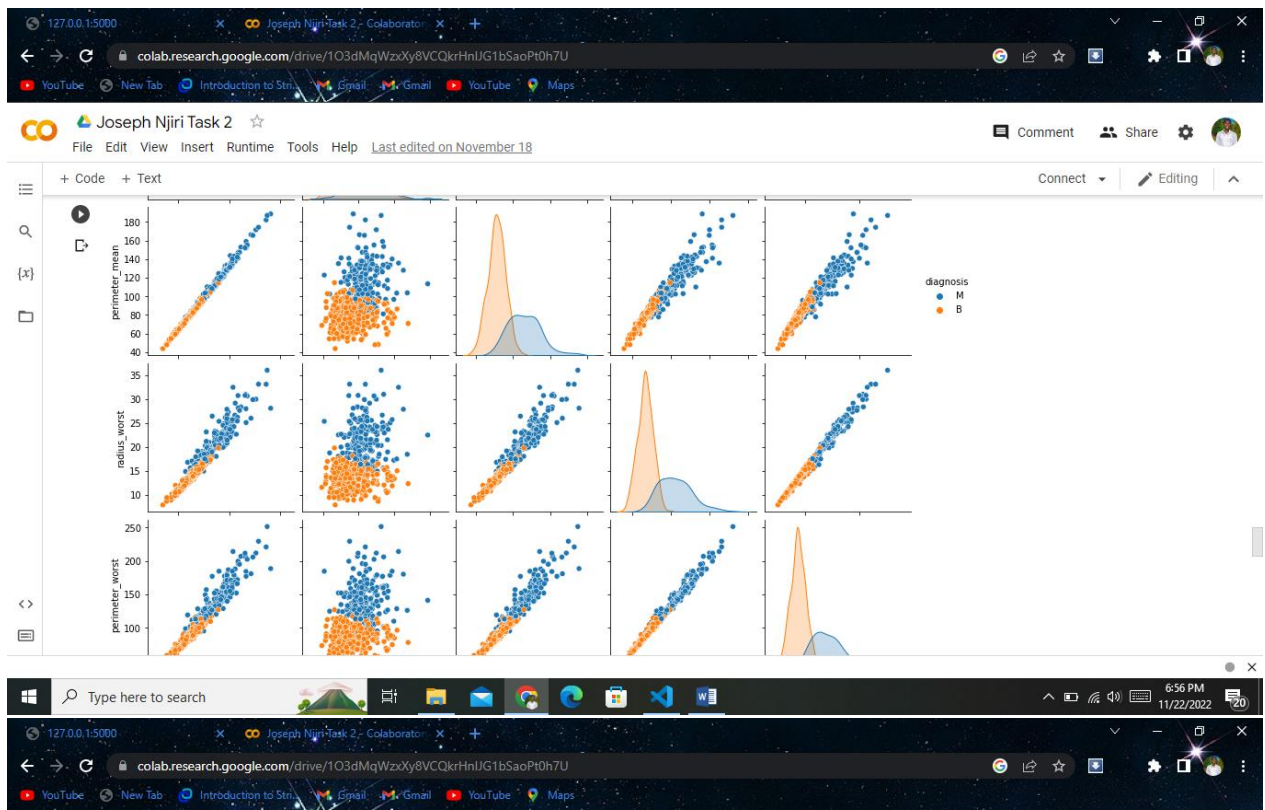


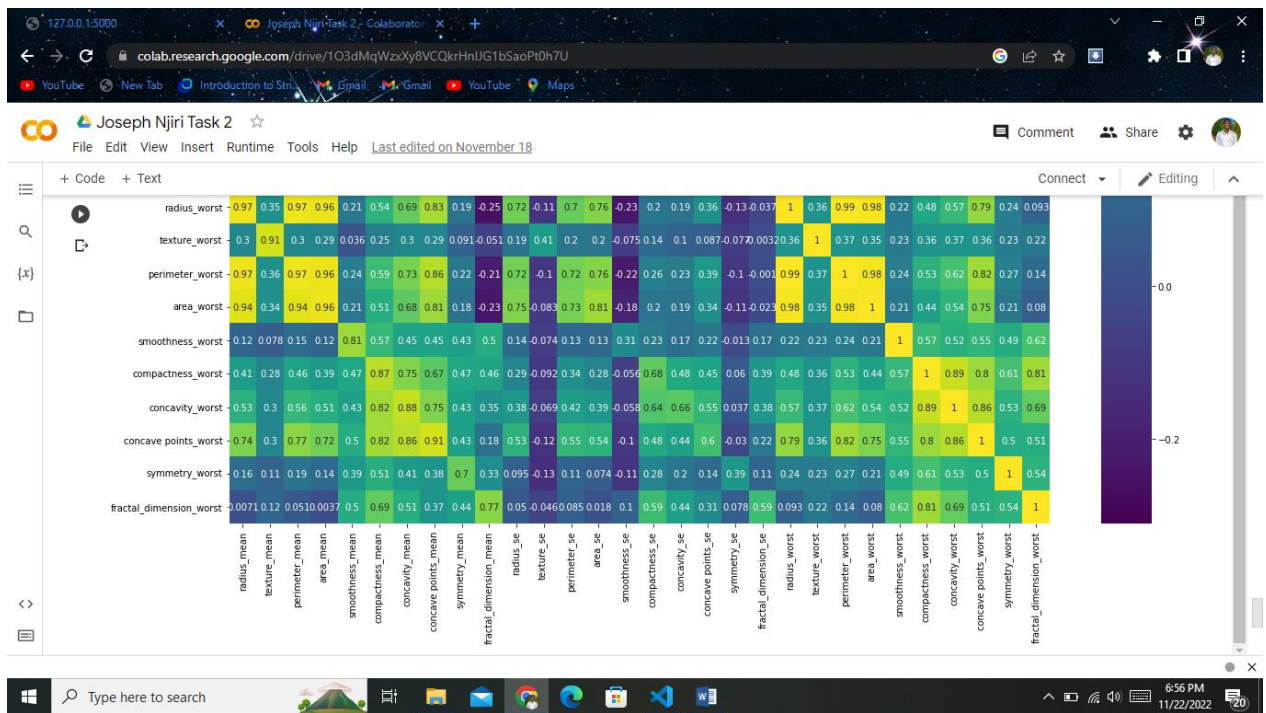
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimension_se	radius_worst	texture_worst
id	int64																							
diagnosis	object																							
radius_mean	float64																							
texture_mean	float64																							
perimeter_mean	float64																							
area_mean	float64																							
smoothness_mean	float64																							
compactness_mean	float64																							
concavity_mean	float64																							
concave points_mean	float64																							
symmetry_mean	float64																							
fractal_dimension_mean	float64																							
radius_se	float64																							
texture_se	float64																							
perimeter_se	float64																							
area_se	float64																							
smoothness_se	float64																							
compactness_se	float64																							
concavity_se	float64																							
concave points_se	float64																							
symmetry_se	float64																							
fractal_dimension_se	float64																							
radius_worst	float64																							
texture_worst	float64																							



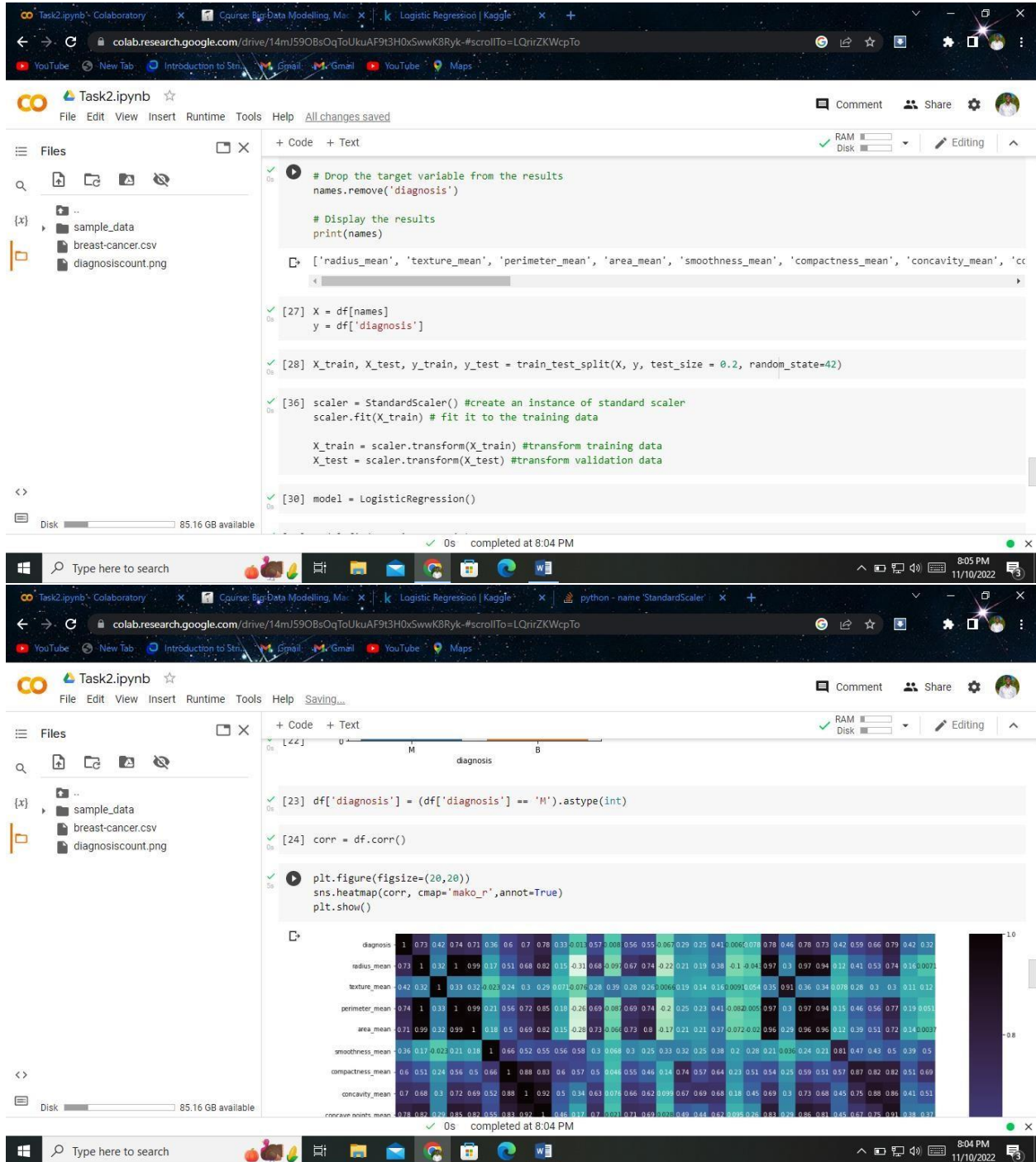








Logistic Regression



The screenshot shows a Google Colaboratory notebook titled 'Task2.ipynb'. The left sidebar displays a file explorer with a folder named 'sample_data' containing 'breast-cancer.csv' and 'diagnosiscount.png'. The main code editor shows the following Python code:

```
[30] X_test = scaler.transform(X_test) #transform validation data
[30] model = LogisticRegression()
[30] model.fit(X_train, y_train)
[32] LogisticRegression()
[32] LogisticRegression()
[32] predictions = model.predict(X_test)
[34] #EVALUATION
[34] accuracy = accuracy_score(y_test, predictions)
[35] print(f'the model accuracy: {accuracy}')
```

The output console at the bottom shows the result of the print statement: 'the model accuracy: 0.9649122807017544'. The notebook interface includes a top bar with 'Task2.ipynb', 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus. The bottom status bar indicates 'completed at 8:04 PM' and '8:05 PM 11/10/2022'.

SUPPORT VECTOR MACHINES(SVM)

(SVM) support vector machine can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well in this case the benign and the malignant type of cancer it gave the highest accuracy of 0.98/98% thus it's the best machine learning solution for the breast cancer dataset. Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyperplane/ line).

Task2.ipynb Colaboratory

colab.research.google.com/drive/14mJ590BsOqToUkuAF9t3H0xSwwK3Ryk-#scrollTo=TBk788HAnJX_

Task2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- sample_data
- breast-cancer.csv
- diagnosiscount.png

```
[42] pred = model.predict(X_test)
```

```
[43] from sklearn.metrics import classification_report, confusion_matrix
```

```
print(classification_report(y_test, pred))
print('\n')
print(confusion_matrix(y_test, pred))
```

	precision	recall	f1-score	support
0	0.97	0.99	0.98	71
1	0.98	0.95	0.96	43
accuracy			0.97	114
macro avg	0.97	0.97	0.97	114
weighted avg	0.97	0.97	0.97	114

```
[[70 1]
 [ 2 41]]
```

```
[45] from sklearn.model_selection import GridSearchCV
```

0s completed at 8:37 PM

Task2.ipynb Colaboratory

colab.research.google.com/drive/14mJ590BsOqToUkuAF9t3H0xSwwK3Ryk-#scrollTo=TBk788HAnJX_

Task2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- sample_data
- breast-cancer.csv
- diagnosiscount.png

```
[45] from sklearn.model_selection import GridSearchCV
```

```
[46] param_grid = {'C':[0.1,1,10,100,1000], 'gamma':[1,0.1,0.01,0.001,0.0001]}
```

```
[47] grid = GridSearchCV(SVC(), param_grid, verbose=3)
```

```
[48] grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END .....C=0.1, gamma=1, score=0.637 total time= 0.0s
[CV 2/5] END .....C=0.1, gamma=1, score=0.626 total time= 0.0s
[CV 3/5] END .....C=0.1, gamma=1, score=0.626 total time= 0.0s
[CV 4/5] END .....C=0.1, gamma=1, score=0.626 total time= 0.0s
[CV 5/5] END .....C=0.1, gamma=1, score=0.626 total time= 0.0s
[CV 1/5] END .....C=0.1, gamma=0.1, score=0.912 total time= 0.0s
[CV 2/5] END .....C=0.1, gamma=0.1, score=0.967 total time= 0.0s
[CV 3/5] END .....C=0.1, gamma=0.1, score=0.934 total time= 0.0s
[CV 4/5] END .....C=0.1, gamma=0.1, score=0.934 total time= 0.0s
[CV 5/5] END .....C=0.1, gamma=0.1, score=0.912 total time= 0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, score=0.934 total time= 0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, score=0.923 total time= 0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, score=0.956 total time= 0.0s
```

0s completed at 8:37 PM

```
macro avg      0.99      0.98      0.98      114
[52] weighted avg  0.98      0.98      0.98      114

[[71  0]
 [ 2 41]]

log = round(grid.score(X,y)*100,2)
print(log)
37.26
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but SVC was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"

[54] from sklearn.metrics import accuracy_score
accuracy_score(y_test, grid_predict)
0.9824561403508771

[ ]
```

NEURAL NETWORKS

```
[ ]
cache = {}
return A, cache

def sigmoid(Z):
    """
    Implement the Sigmoid function.

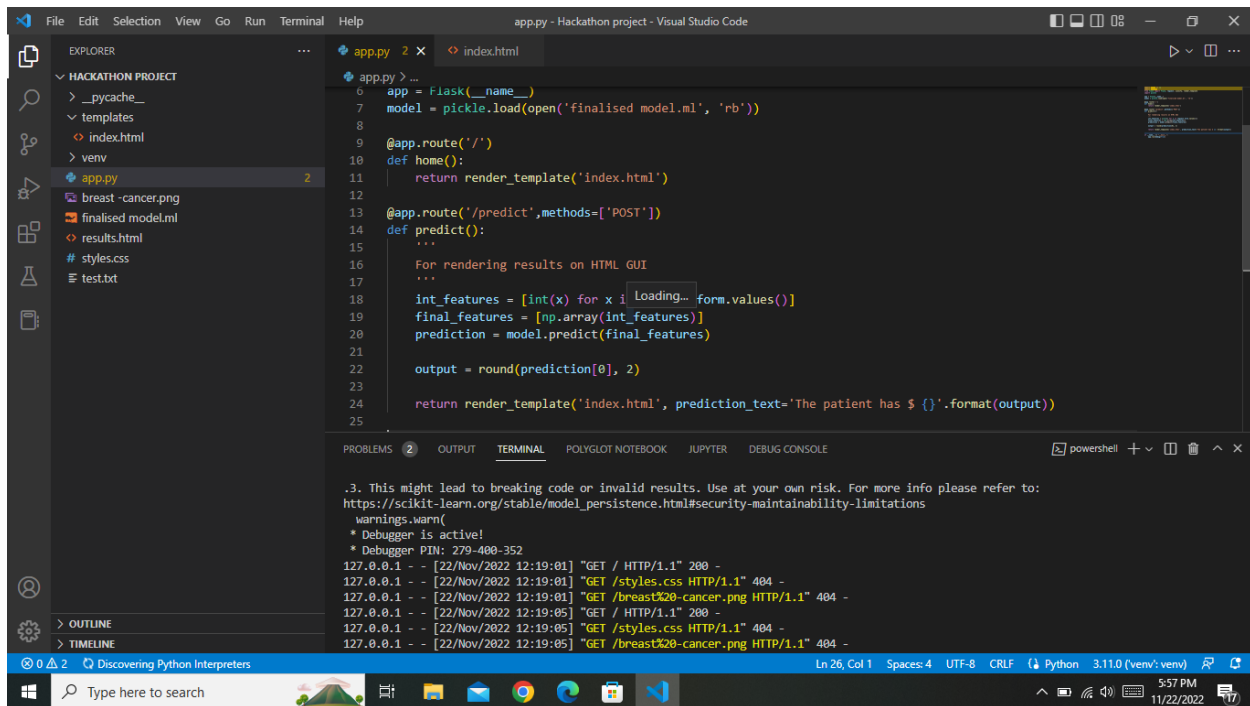
    Arguments:
    Z -- Output of the linear layer

    Returns:
    A -- Post-activation parameter
    cache -- a python dictionary containing "A" for backpropagation
    """
    A = 1/(1+np.exp(-Z))
    cache = Z
    return A, cache

[ ] z = np.linspace(-12, 12, 200)
fig = px.line(x=z, y=sigmoid(z)[0],title='Sigmoid Function',template="plotly_dark")
fig.update_layout(
    title_font_color="#00FFFF",
    xaxis=dict(color="#00FFFF"),
    yaxis=dict(color="#00FFFF")
)
```

The above machine learning technique gives an accuracy of 0.96. It evaluates the model using the sigmoid function and creating a class neural networks. The process of model evaluation was too long but it gave a lower accuracy compared to support vector machine algorithm svm.

Model Deployment



The screenshot shows the Visual Studio Code interface with a project named 'app.py - Hackathon project'. The Explorer panel on the left shows the project structure, including files like 'breast-cancer.png', 'finalised model.pkl', 'results.html', 'styles.css', and 'test.txt'. The main editor displays the 'app.py' file, which is a Flask application. The code defines a Flask app, loads a pickle model, and sets up two routes: a home route and a predict route. The predict route takes a POST request, processes the input features, and returns a prediction. The terminal at the bottom shows the output of the application, including a warning about model persistence and a list of HTTP requests and responses.

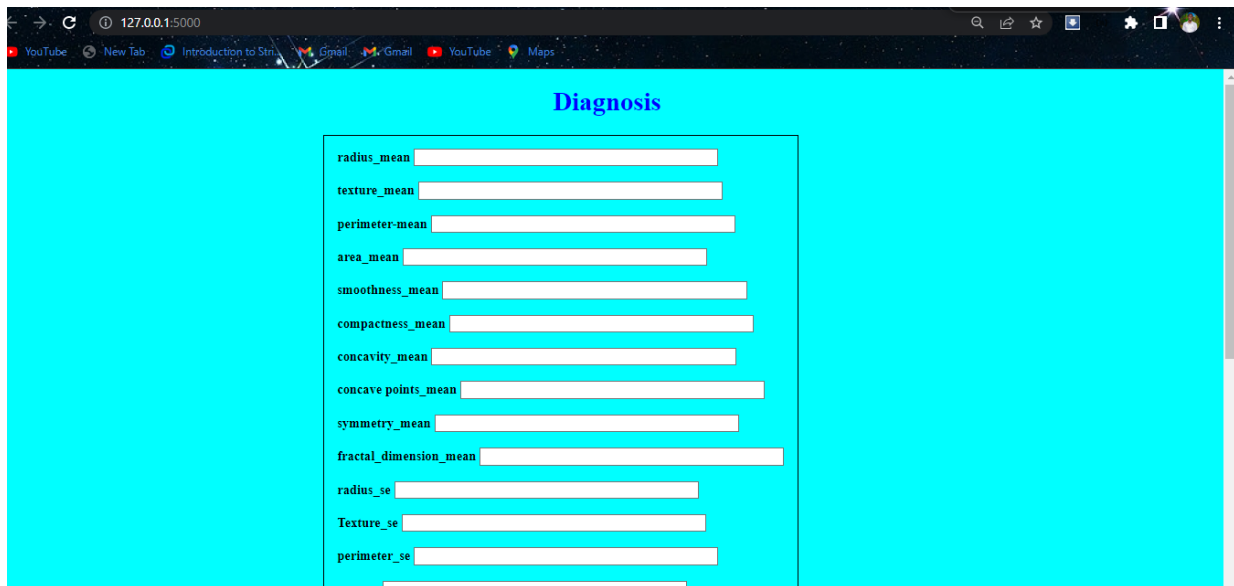
```
app.py 2 x index.html
app.py > ...
6 app = Flask(__name__)
7 model = pickle.load(open('finalised model.pkl', 'rb'))
8
9 @app.route('/')
10 def home():
11     return render_template('index.html')
12
13 @app.route('/predict', methods=['POST'])
14 def predict():
15     ...
16     For rendering results on HTML GUI
17     ...
18     int_features = [int(x) for x in request.form.values()]
19     final_features = [np.array(int_features)]
20     prediction = model.predict(final_features)
21
22     output = round(prediction[0], 2)
23
24     return render_template('index.html', prediction_text='The patient has {}.'.format(output))
25
```

PROBLEMS 2 OUTPUT TERMINAL POLYGLOT NOTEBOOK JUPYTER DEBUG CONSOLE

.3. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations

warnings.warn(
 * Debugger is active!
 * Debugger PIN: 279-480-352

127.0.0.1 - - [22/Nov/2022 12:19:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2022 12:19:01] "GET /styles.css HTTP/1.1" 404 -
127.0.0.1 - - [22/Nov/2022 12:19:01] "GET /breast-cancer.png HTTP/1.1" 404 -
127.0.0.1 - - [22/Nov/2022 12:19:05] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2022 12:19:05] "GET /styles.css HTTP/1.1" 404 -
127.0.0.1 - - [22/Nov/2022 12:19:05] "GET /breast-cancer.png HTTP/1.1" 404 -



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000'. The browser has several tabs open, including 'YouTube', 'New Tab', 'Introduction to Str...', 'Gmail', and 'Maps'. The main content of the browser is a web page titled 'Diagnosis'. The page has a light blue background and contains a form with several input fields. The form is titled 'Diagnosis' and has a light blue border. The input fields are labeled with various features: 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean', 'radius_se', 'Texture_se', and 'perimeter_se'. Each label is followed by an input field. The form is currently empty, and the user is expected to enter values into these fields.

smoothness_se

compactness_se

concavity_se

concave points_se

symmetry_se

fractal_dimension_se

radius_worst

texture_worst

perimeter_worst

area_worst

smoothness_worst

compactness_worst

concavity_worst

concave points_worst

symmetry_worst

fractal_dimension_worst

Predict

Given breast cancer results from breast fine needle aspiration (FNA) test (is a quick and simple procedure to perform, which removes some fluid or cells from a breast lesion or cyst (a lump, sore or swelling) with a fine needle similar to a blood sample needle). Since this build a model that can classify a breast cancer tumor using two training classification:

1= Malignant (Cancerous) - Present

0= Benign (Not Cancerous) -Absent

After inputing different types of data a diagnosis will be given in form of text showing whether the cancer is malignant(M) or benign(B).