Machine Learning

(Practical File)

Submitted By: Jaskaran Singh

Enrollment No.: 05113202717

Class: CSE-01

AIM:

Estimate the accuracy of the decision classifier on breast cancer dataset using 5 fold cross validation.

ALGORITHM:

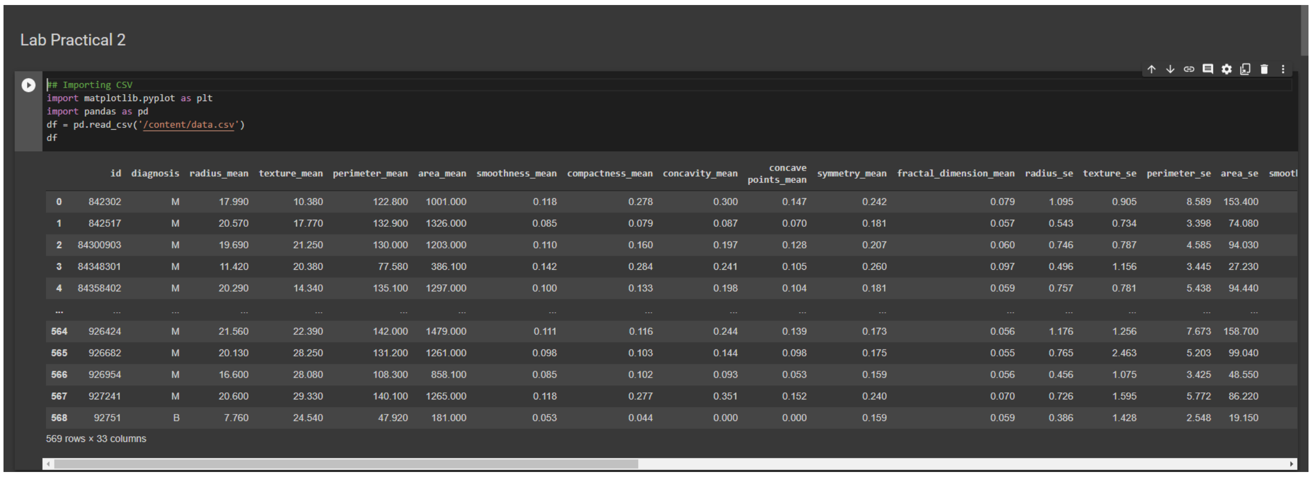
1. Reserve some portion of sample data-set.
2. Using the rest data-set train the model.
3. Test the model using the reserve portion of the data-set.

K-Fold Cross Validation

In this method, we split the data-set into k number of subsets(known as folds) then we perform training on all the subsets but leave one(k-1) subset for the evaluation of the trained model. In this method, we iterate k times with a different subset reserved for testing purposes each time.

PROGRAM CODE SNIPPET:

LOADING DATA SET:



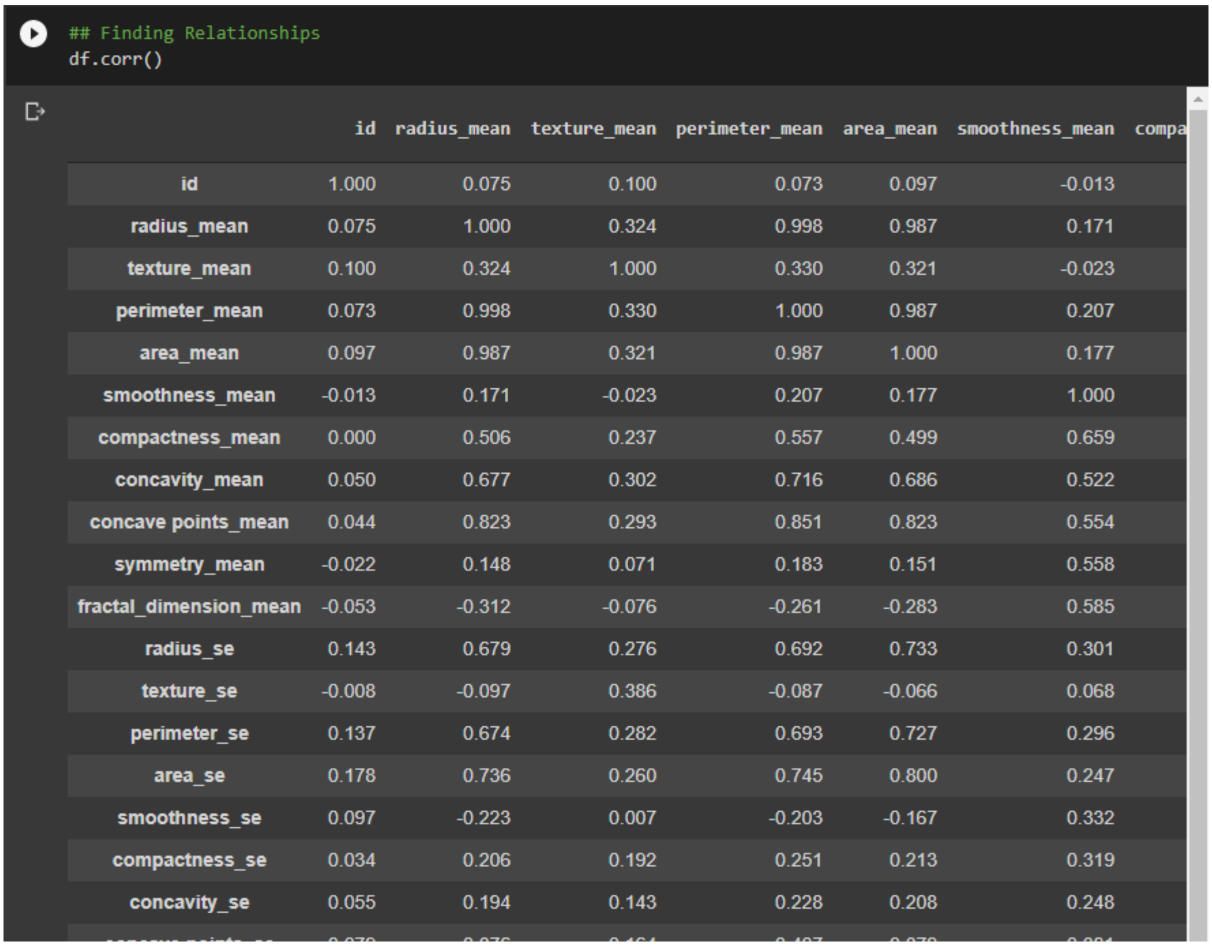
Lab Practical 2

```
## Importing CSV
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('/content/data.csv')
df
```

	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	smoot
0	842302	M	17.990	10.380	122.800	1001.000	0.118	0.278	0.300	0.147	0.242	0.079	1.095	0.905	8.589	153.400	
1	842517	M	20.570	17.770	132.900	1326.000	0.085	0.079	0.087	0.070	0.181	0.057	0.543	0.734	3.398	74.080	
2	84300903	M	19.690	21.250	130.000	1203.000	0.110	0.160	0.197	0.128	0.207	0.060	0.746	0.787	4.585	94.030	
3	84348301	M	11.420	20.380	77.580	386.100	0.142	0.284	0.241	0.105	0.260	0.097	0.496	1.156	3.445	27.230	
4	84358402	M	20.290	14.340	135.100	1297.000	0.100	0.133	0.198	0.104	0.181	0.059	0.757	0.781	5.438	94.440	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	926424	M	21.560	22.390	142.000	1479.000	0.111	0.116	0.244	0.139	0.173	0.056	1.176	1.256	7.673	158.700	
565	926682	M	20.130	28.250	131.200	1261.000	0.098	0.103	0.144	0.098	0.175	0.055	0.765	2.463	5.203	99.040	
566	926954	M	16.600	28.080	108.300	858.100	0.085	0.102	0.093	0.053	0.159	0.056	0.456	1.075	3.425	48.550	
567	927241	M	20.600	29.330	140.100	1265.000	0.118	0.277	0.351	0.152	0.240	0.070	0.726	1.595	5.772	86.220	
568	92751	B	7.760	24.540	47.920	181.000	0.053	0.044	0.000	0.000	0.159	0.059	0.386	1.428	2.548	19.150	

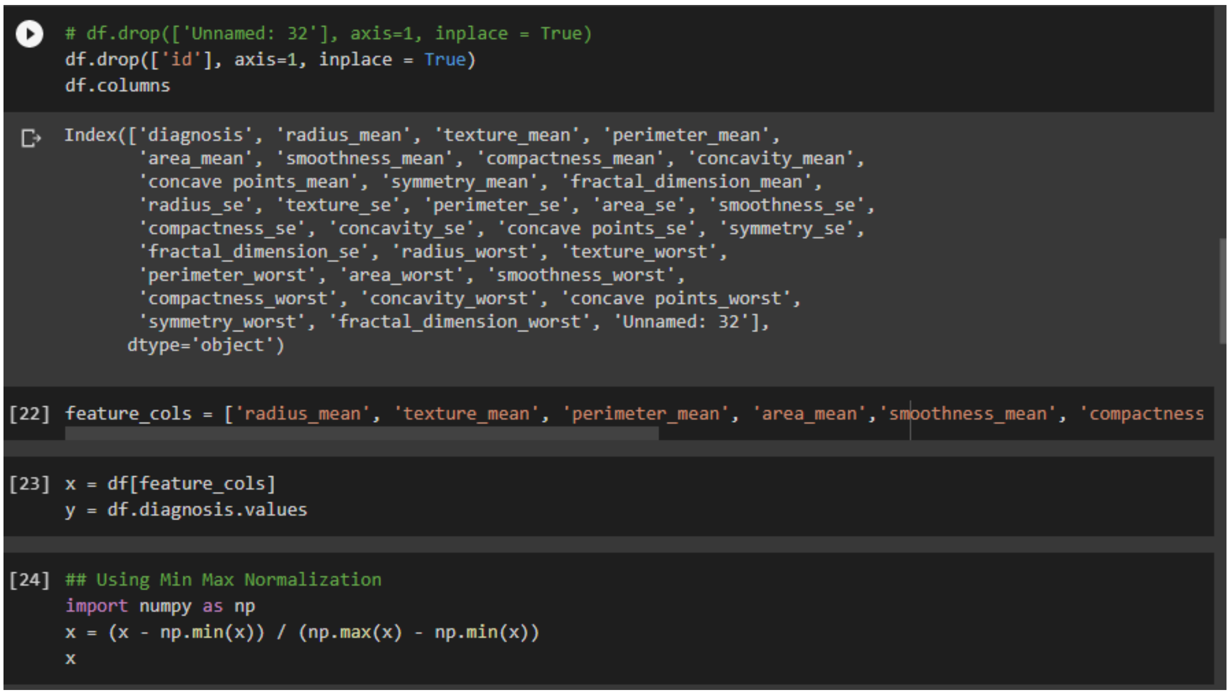
569 rows × 33 columns

PREPROCESSING:



```
## Finding Relationships
df.corr()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compa
id	1.000	0.075	0.100	0.073	0.097	-0.013	
radius_mean	0.075	1.000	0.324	0.998	0.987	0.171	
texture_mean	0.100	0.324	1.000	0.330	0.321	-0.023	
perimeter_mean	0.073	0.998	0.330	1.000	0.987	0.207	
area_mean	0.097	0.987	0.321	0.987	1.000	0.177	
smoothness_mean	-0.013	0.171	-0.023	0.207	0.177	1.000	
compactness_mean	0.000	0.506	0.237	0.557	0.499	0.659	
concavity_mean	0.050	0.677	0.302	0.716	0.686	0.522	
concave points_mean	0.044	0.823	0.293	0.851	0.823	0.554	
symmetry_mean	-0.022	0.148	0.071	0.183	0.151	0.558	
fractal_dimension_mean	-0.053	-0.312	-0.076	-0.261	-0.283	0.585	
radius_se	0.143	0.679	0.276	0.692	0.733	0.301	
texture_se	-0.008	-0.097	0.386	-0.087	-0.066	0.068	
perimeter_se	0.137	0.674	0.282	0.693	0.727	0.296	
area_se	0.178	0.736	0.260	0.745	0.800	0.247	
smoothness_se	0.097	-0.223	0.007	-0.203	-0.167	0.332	
compactness_se	0.034	0.206	0.192	0.251	0.213	0.319	
concavity_se	0.055	0.194	0.143	0.228	0.208	0.248	



```
# df.drop(['Unnamed: 32'], axis=1, inplace = True)
df.drop(['id'], axis=1, inplace = True)
df.columns
```

```
Index(['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',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

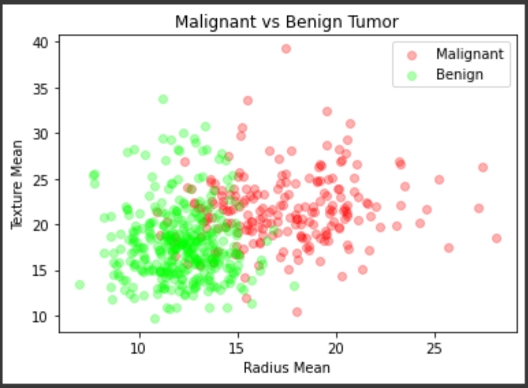
```
[22] feature_cols = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean','smoothness_mean', 'compactness
```

```
[23] x = df[feature_cols]
     y = df.diagnosis.values
```

```
[24] ## Using Min Max Normalization
     import numpy as np
     x = (x - np.min(x)) / (np.max(x) - np.min(x))
     x
```

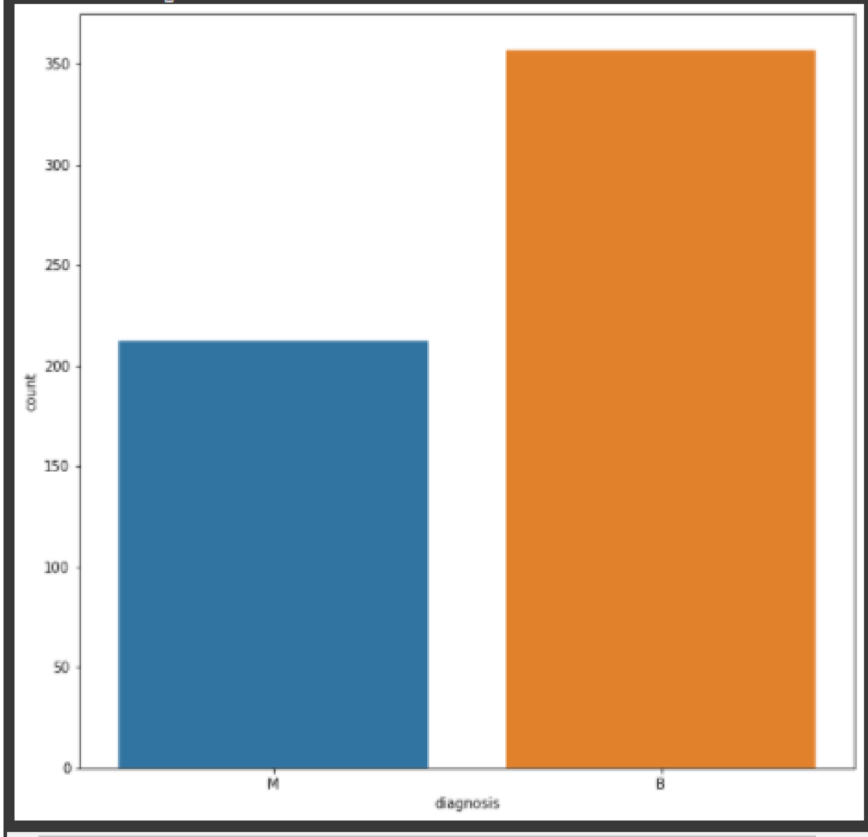
VISUALIZATION:

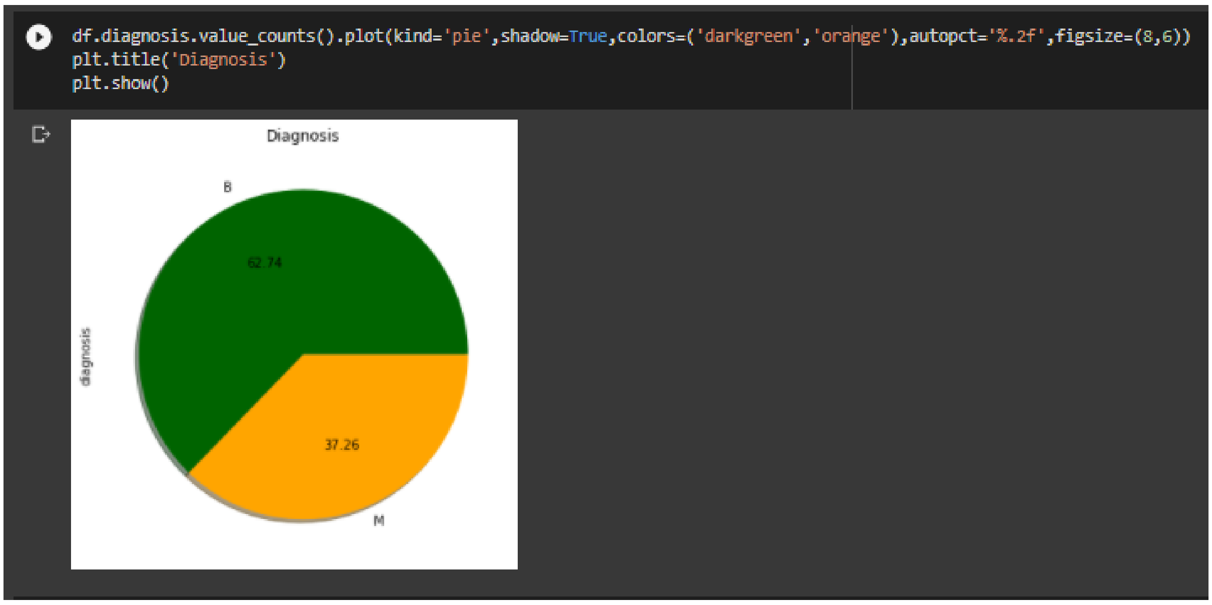
```
M = df[df.diagnosis == "M"]

B = df[df.diagnosis == "B"]

plt.title("Malignant vs Benign Tumor")
plt.xlabel("Radius Mean")
plt.ylabel("Texture Mean")
plt.scatter(M.radius_mean, M.texture_mean, color = "red", label = "Malignant", alpha = 0.3)
plt.scatter(B.radius_mean, B.texture_mean, color = "lime", label = "Benign", alpha = 0.3)
plt.legend()
plt.show()
```



```
plt.figure(figsize=(10,10))
sns.countplot(df['diagnosis'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
  FutureWarning
```

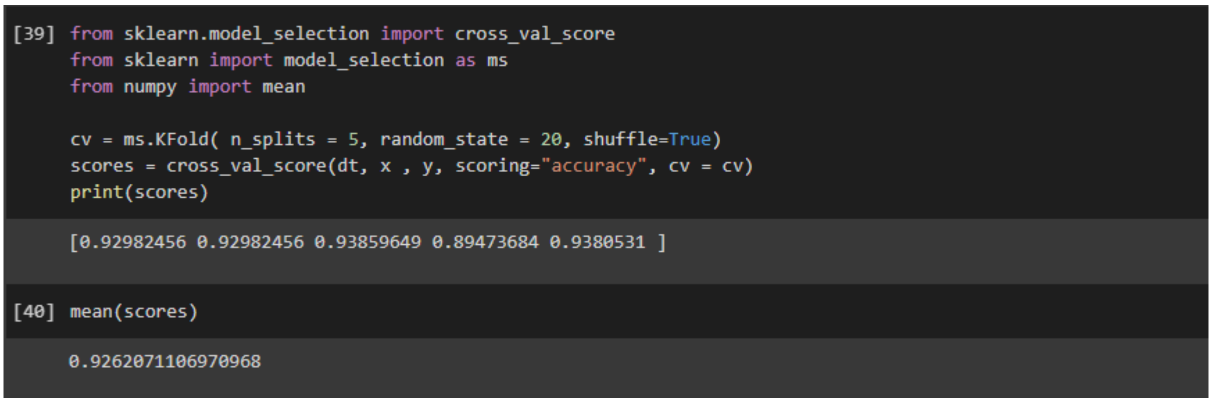




```
df.diagnosis.value_counts().plot(kind='pie',shadow=True,colors=('darkgreen','orange'),autopct='%.2f',figsize=(8,6))
plt.title('Diagnosis')
plt.show()
```

ML

ALGORITHM IMPLEMENTATION:



```
[39] from sklearn.model_selection import cross_val_score
     from sklearn import model_selection as ms
     from numpy import mean

     cv = ms.KFold( n_splits = 5, random_state = 20, shuffle=True)
     scores = cross_val_score(dt, x , y, scoring="accuracy", cv = cv)
     print(scores)

     [0.92982456 0.92982456 0.93859649 0.89473684 0.9380531 ]

[40] mean(scores)

     0.9262071106970968
```

GITHUB LINK:

https://github.com/jaskarans2000/Python-Introduction-Lab---Assignment-1---Jaskaran-Singh/LabAssignment3.ipynb