

MACHINE LEARNING FILE

NAME: SHRUTI GUPTA

CLASS: CSE-3

ROLL NO.: 41476802717

EXPERIMENT – 4

Problem Statement:

Estimate the precision recall accuracy f-measure of the decision classifier on a breast cancer dataset using 10-Fold cross validation

Github Link:

<https://github.com/Shrutiig/ML-Assignmnets-Gtbit/tree/master/Lab-4>

Algorithm Description:

Decision Tree: A Decision Tree is a supervised Machine learning algorithm. It is used in both classification and regression algorithms. The decision tree is like a tree with nodes. The branches depend on a number of factors. It splits data into branches like these till it achieves a threshold value. A decision tree consists of the root nodes, children nodes, and leaf nodes.

K-Fold Means: K-fold cross validation is one way to improve the holdout method. This method guarantees that the score of our model does not depend on the way we picked the train and test set. The data set is divided into k number of subsets and the holdout method is repeated k number of times.

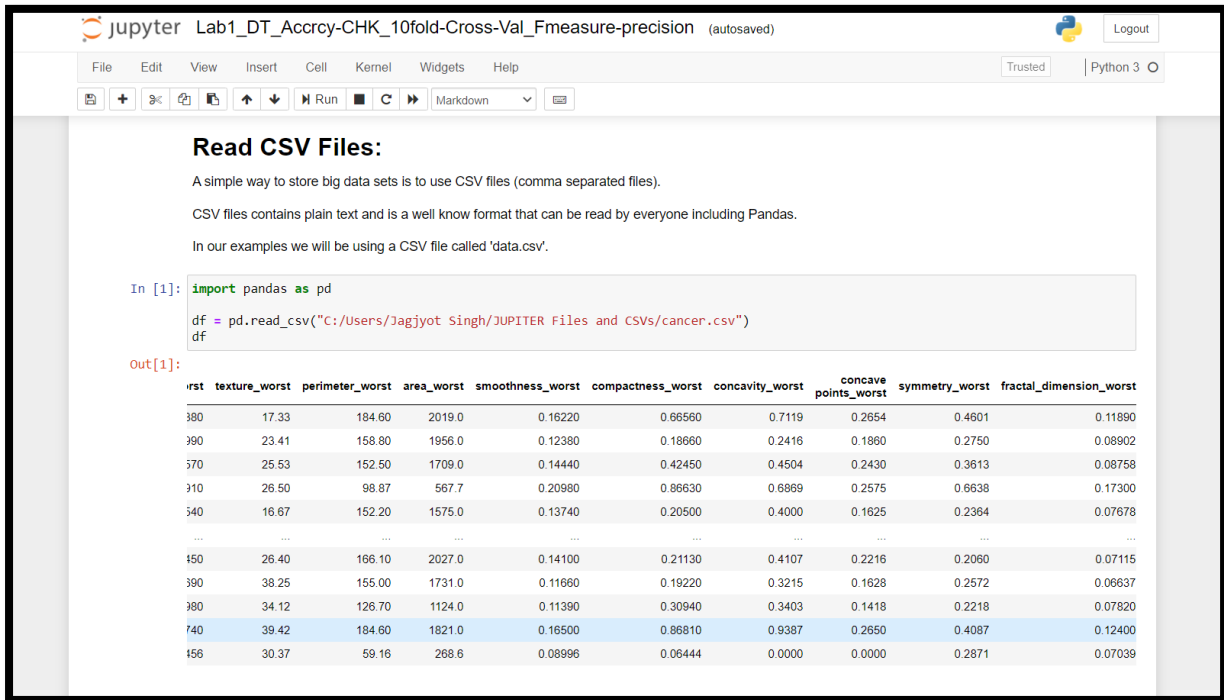
Precision: The ratio $tp / (tp + fp)$ where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

Recall: The ratio $tp / (tp + fn)$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

Accuracy: Defined as the ratio of correctly predicted examples by the total examples.

Program Code Snippet:

Loading dataset:



Read CSV Files:

A simple way to store big data sets is to use CSV files (comma separated files).

CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

In our examples we will be using a CSV file called 'data.csv'.

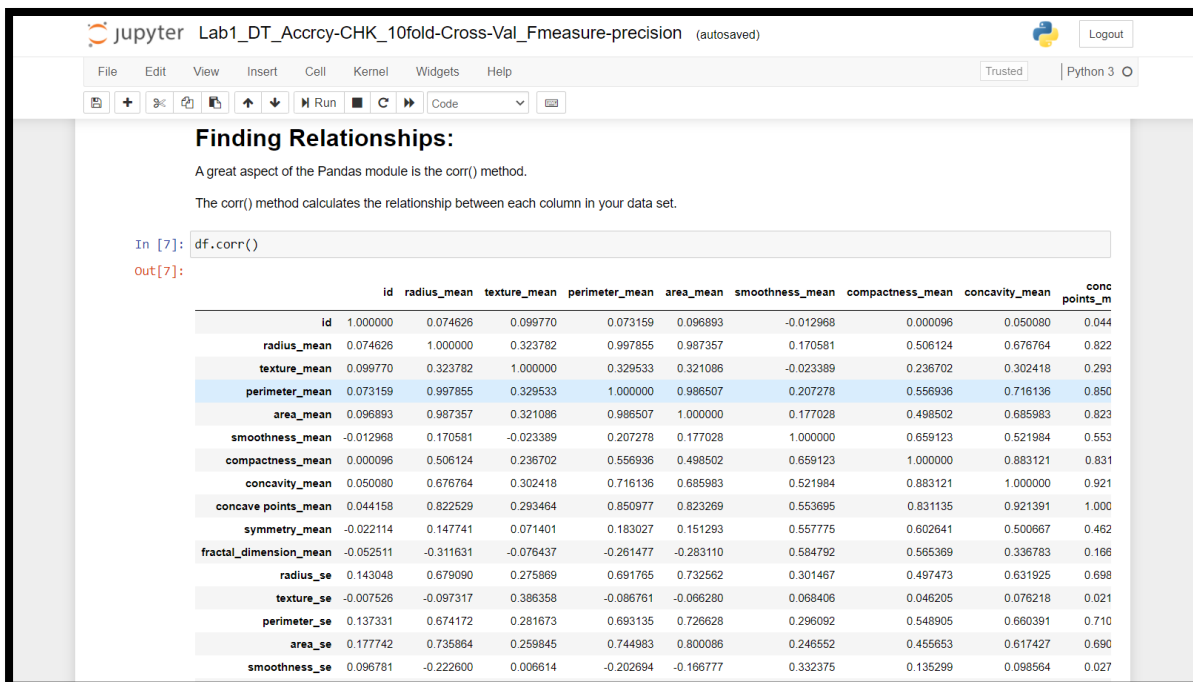
```
In [1]: import pandas as pd

df = pd.read_csv("C:/Users/Jaggyot Singh/JUPITER Files and CSVs/cancer.csv")
df
```

Out[1]:

	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.4601	0.11890
390	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.2750	0.08902
370	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.3613	0.08758
310	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.6638	0.17300
340	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.2364	0.07678
...
150	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115
380	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637
380	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820
740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400
156	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039

Pre-processing of dataset:



Finding Relationships:

A great aspect of the Pandas module is the corr() method.

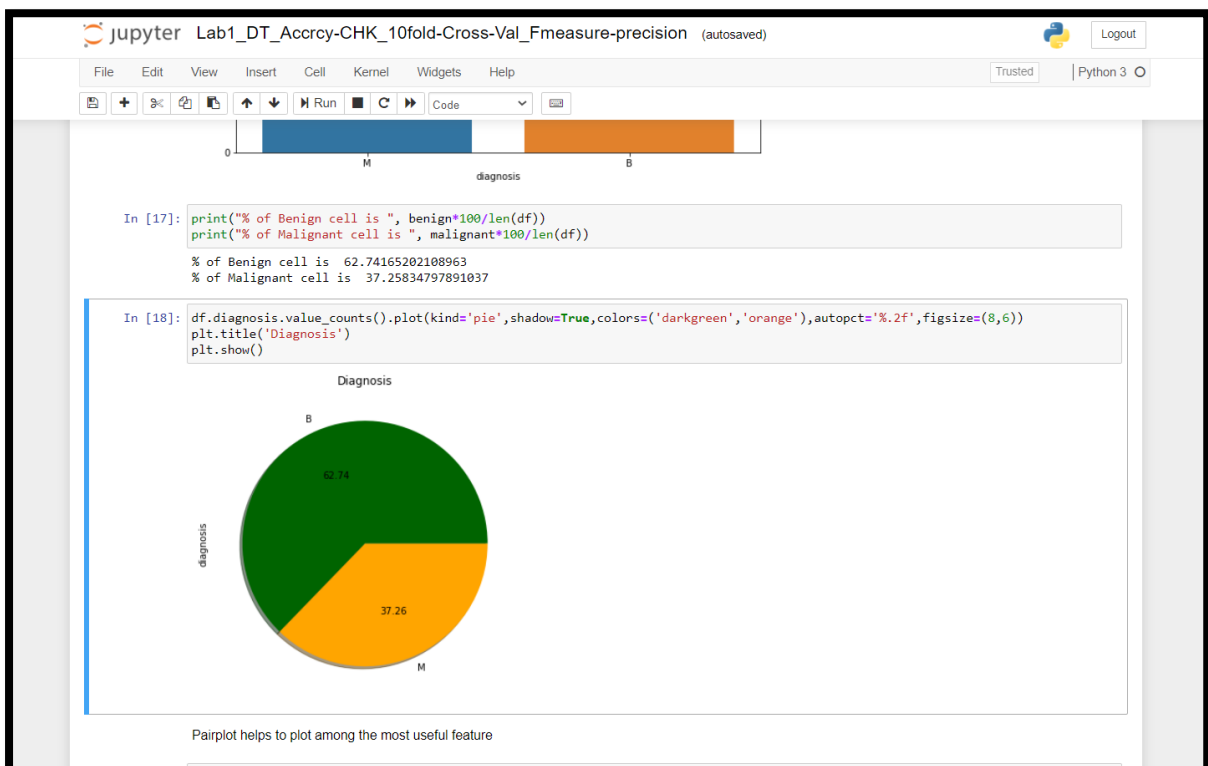
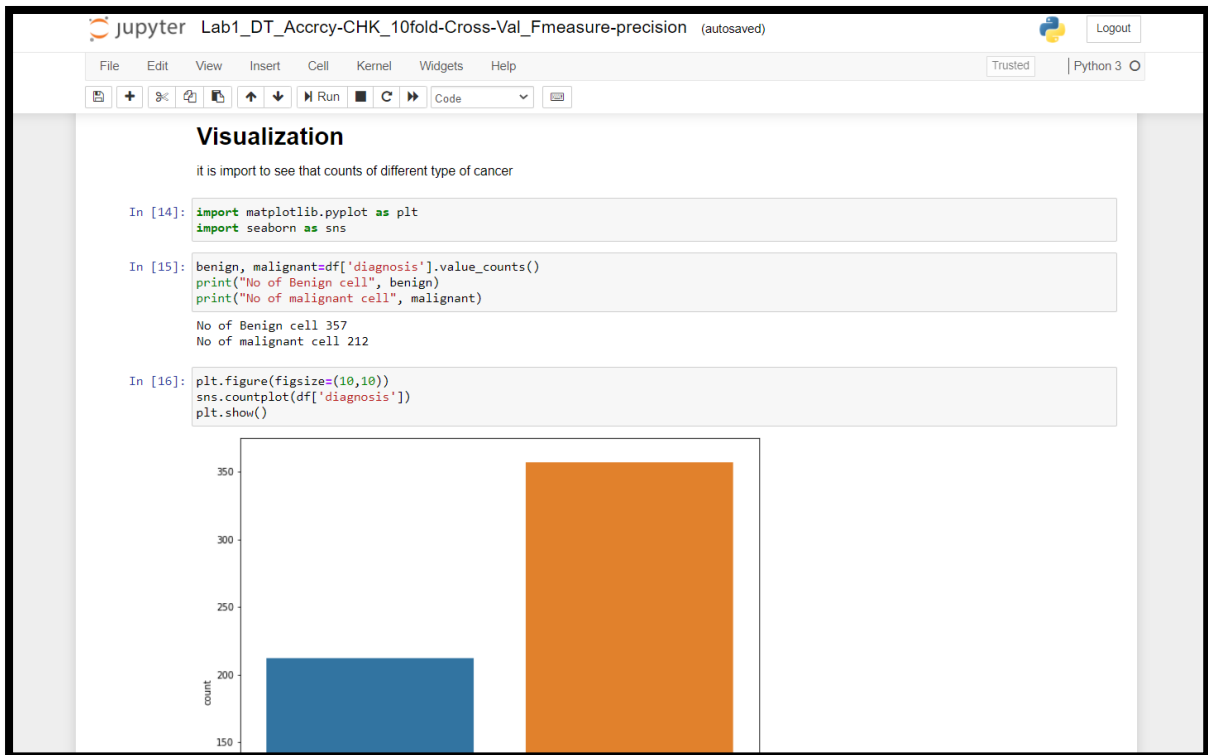
The corr() method calculates the relationship between each column in your data set.

```
In [7]: df.corr()
```

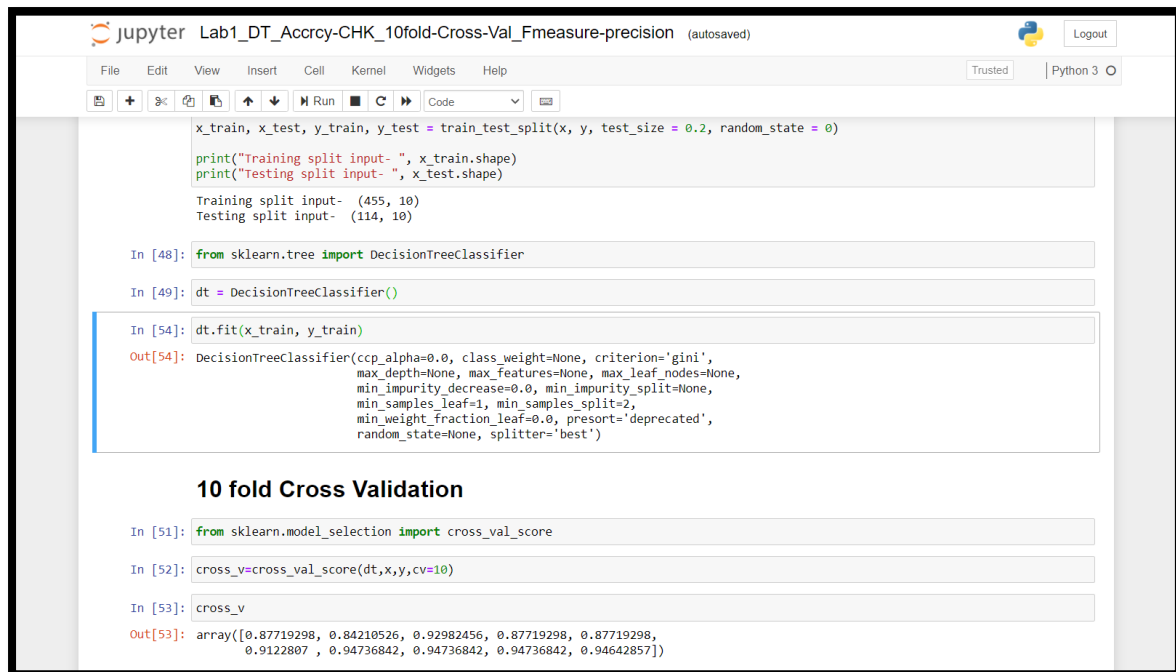
Out[7]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conc points_m
id	1.000000	0.074626	0.099770	0.073159	0.096893	-0.012968	0.000096	0.050080	0.044
radius_mean	0.074626	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822
texture_mean	0.099770	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293
perimeter_mean	0.073159	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850
area_mean	0.096893	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823
smoothness_mean	-0.012968	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553
compactness_mean	0.000096	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831
concavity_mean	0.050080	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921
concave points_mean	0.044158	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000
symmetry_mean	-0.022114	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462
fractal_dimension_mean	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166
radius_se	0.143048	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.698
texture_se	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.021
perimeter_se	0.137331	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.710
area_se	0.177742	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.617427	0.690
smoothness_se	0.096781	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.027

Visualization:



Decision Tree Classifier& 10-Fold Cross Validation:



The image shows a Jupyter Notebook titled "Lab1_DT_Accrcy-CHK_10fold-Cross-Val_Fmeasure-precision" with the following code and output:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)

print("Training split input- ", x_train.shape)
print("Testing split input- ", x_test.shape)

Training split input- (455, 10)
Testing split input- (114, 10)
```

In [48]: `from sklearn.tree import DecisionTreeClassifier`

In [49]: `dt = DecisionTreeClassifier()`

In [54]: `dt.fit(x_train, y_train)`

Out[54]: `DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort='deprecated', random_state=None, splitter='best')`

10 fold Cross Validation

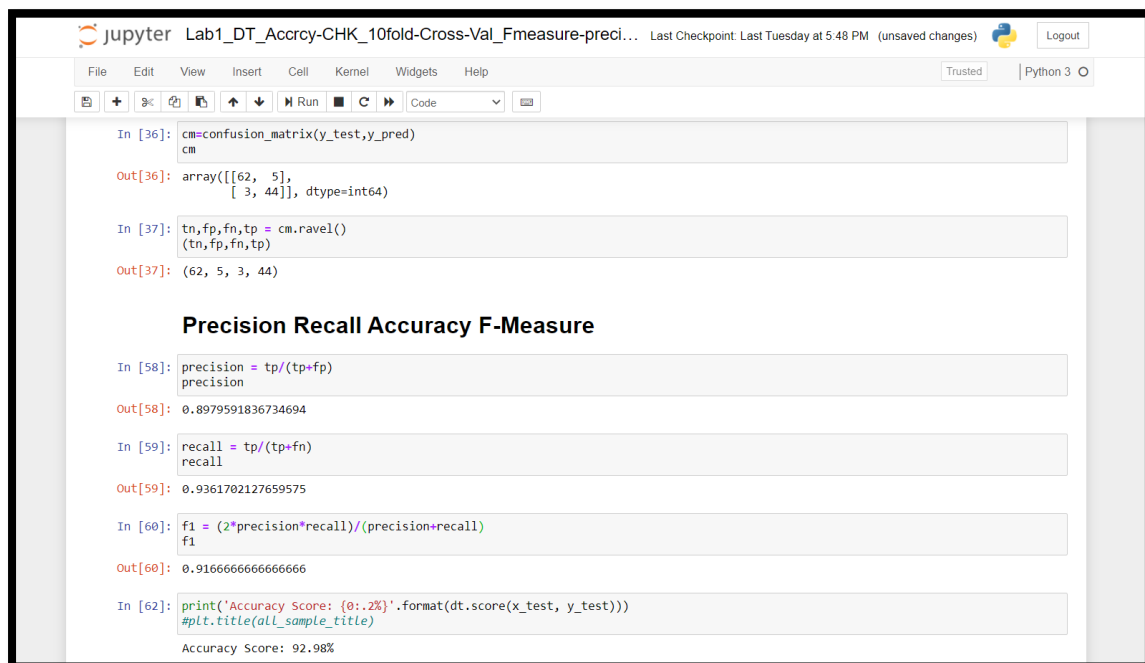
In [51]: `from sklearn.model_selection import cross_val_score`

In [52]: `cross_v=cross_val_score(dt,x,y,cv=10)`

In [53]: `cross_v`

Out[53]: `array([0.87719298, 0.84210526, 0.92982456, 0.87719298, 0.87719298, 0.91228807, 0.94736842, 0.94736842, 0.94736842, 0.94642857])`

Precision Recall Accuracy and F measure:



The image shows a Jupyter Notebook titled "Lab1_DT_Accrcy-CHK_10fold-Cross-Val_Fmeasure-precision" with the following code and output:

```
In [36]: cm=confusion_matrix(y_test,y_pred)
cm

Out[36]: array([[62,  5],
               [ 3, 44]], dtype=int64)
```

In [37]: `tn,fp,fn,tp = cm.ravel()`
`(tn,fp,fn,tp)`

Out[37]: `(62, 5, 3, 44)`

Precision Recall Accuracy F-Measure

In [58]: `precision = tp/(tp+fp)`
`precision`

Out[58]: `0.8979591836734694`

In [59]: `recall = tp/(tp+fn)`
`recall`

Out[59]: `0.9361702127659575`

In [60]: `f1 = (2*precision*recall)/(precision+recall)`
`f1`

Out[60]: `0.9166666666666666`

In [62]: `print('Accuracy Score: {0:.2%}'.format(dt.score(x_test, y_test)))`
`#plt.title(all_sample_title)`

Accuracy Score: 92.98%

Final Graph

