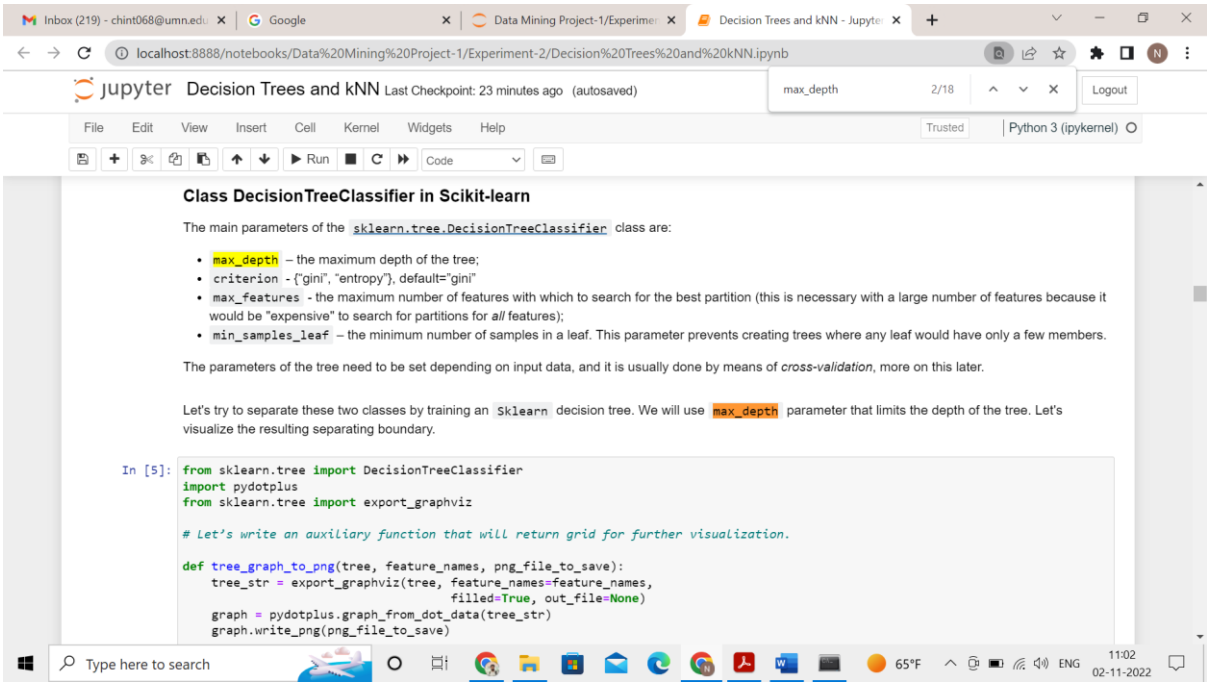
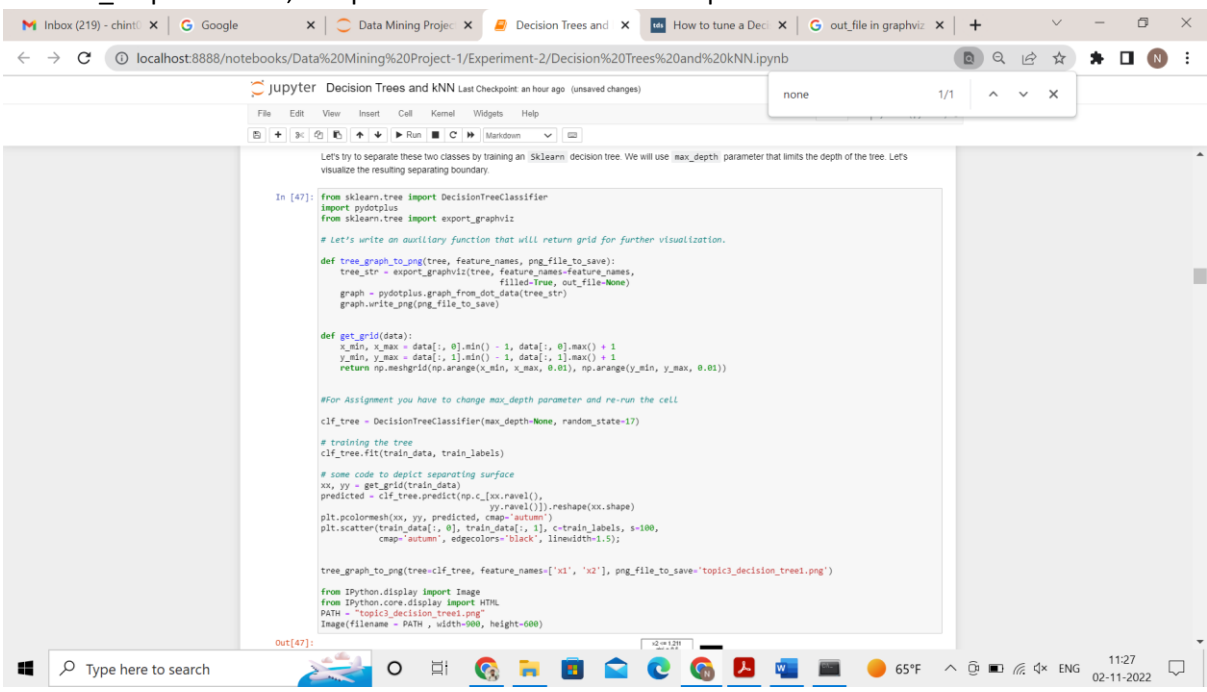
Project - 1

1. For criterion if no value is mentioned then by default Gini is considered.



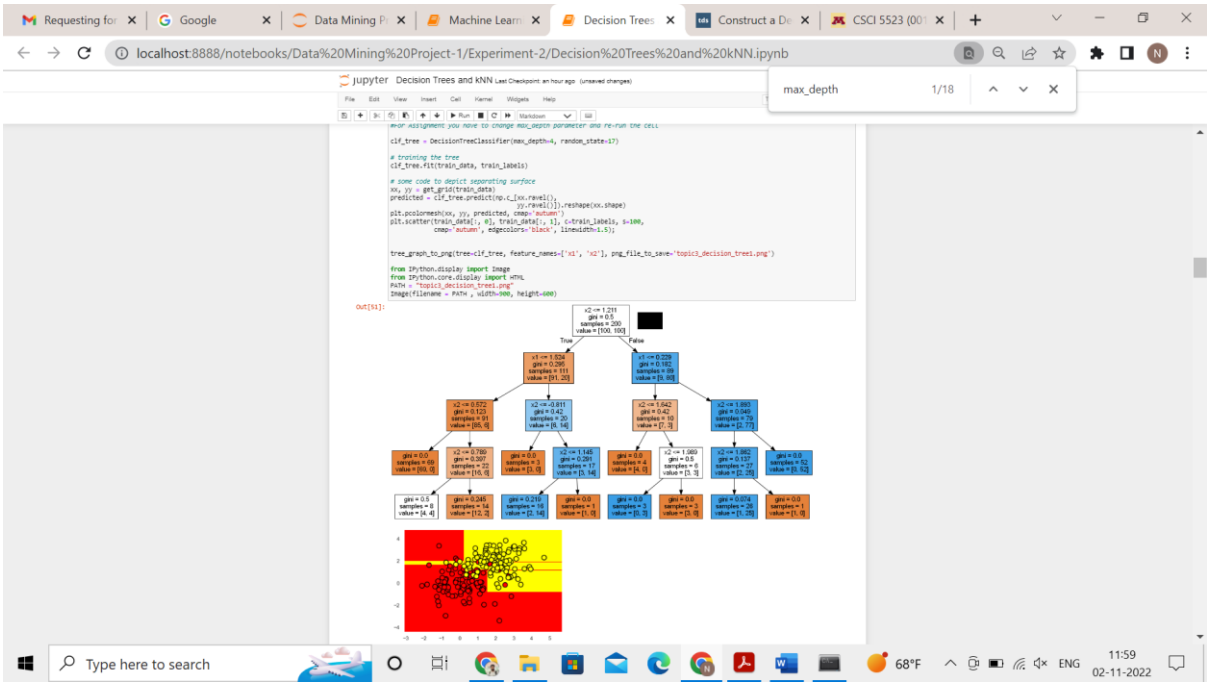
For max_ depth is None, it expands tree until all leaves are pure.



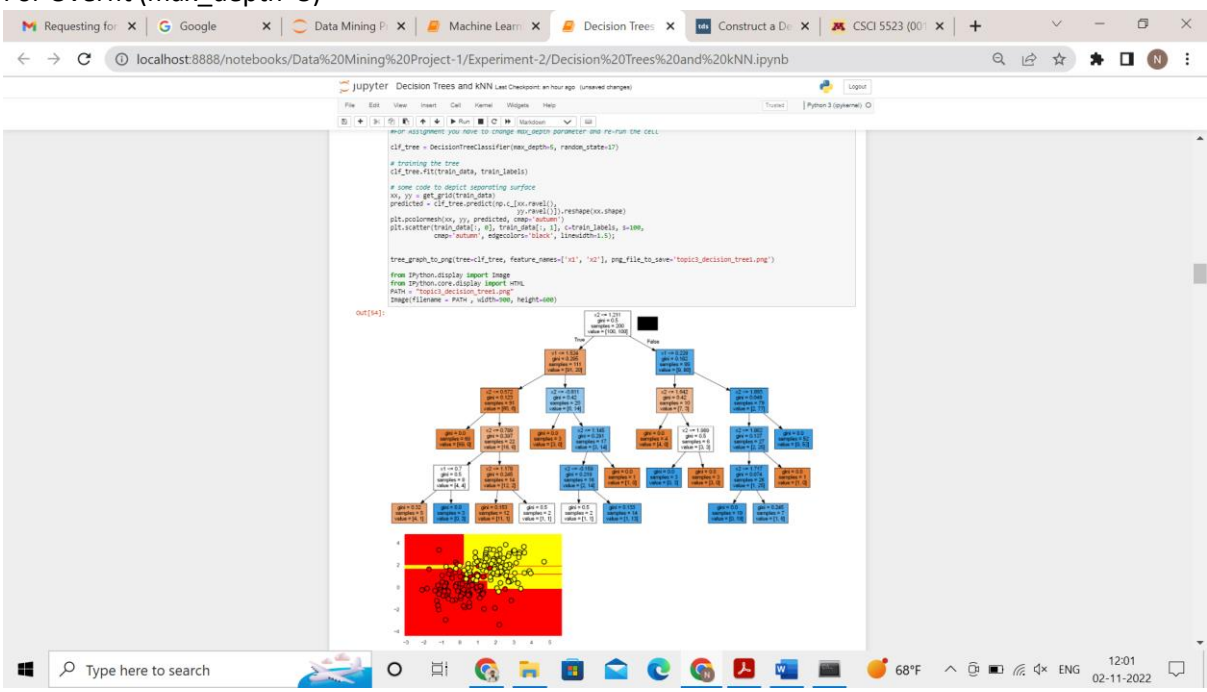
2.

For Overfit (max_depth=4)

Project - 1

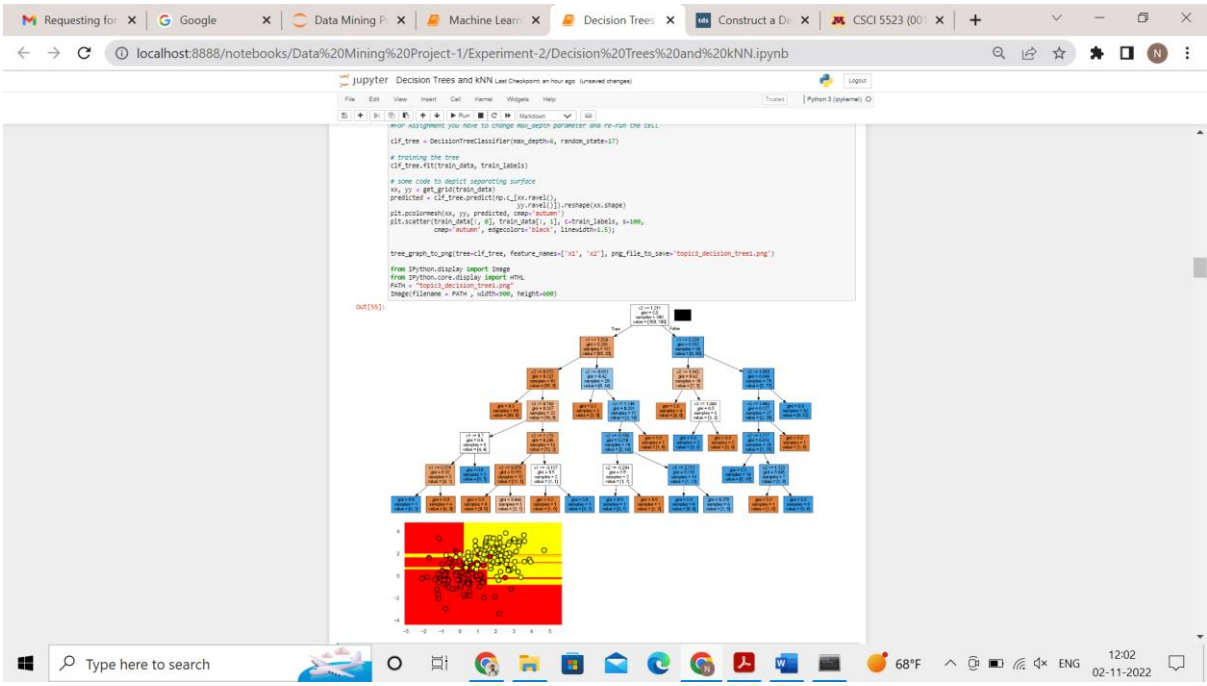


For Overfit (max_depth=5)

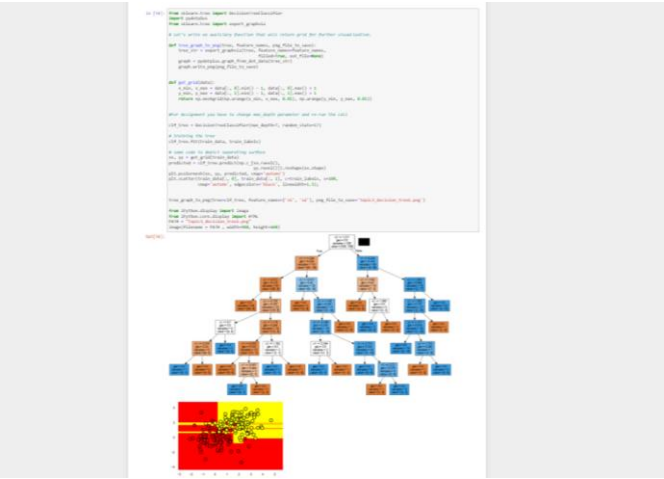


For Overfit (max_depth=6)

Project - 1

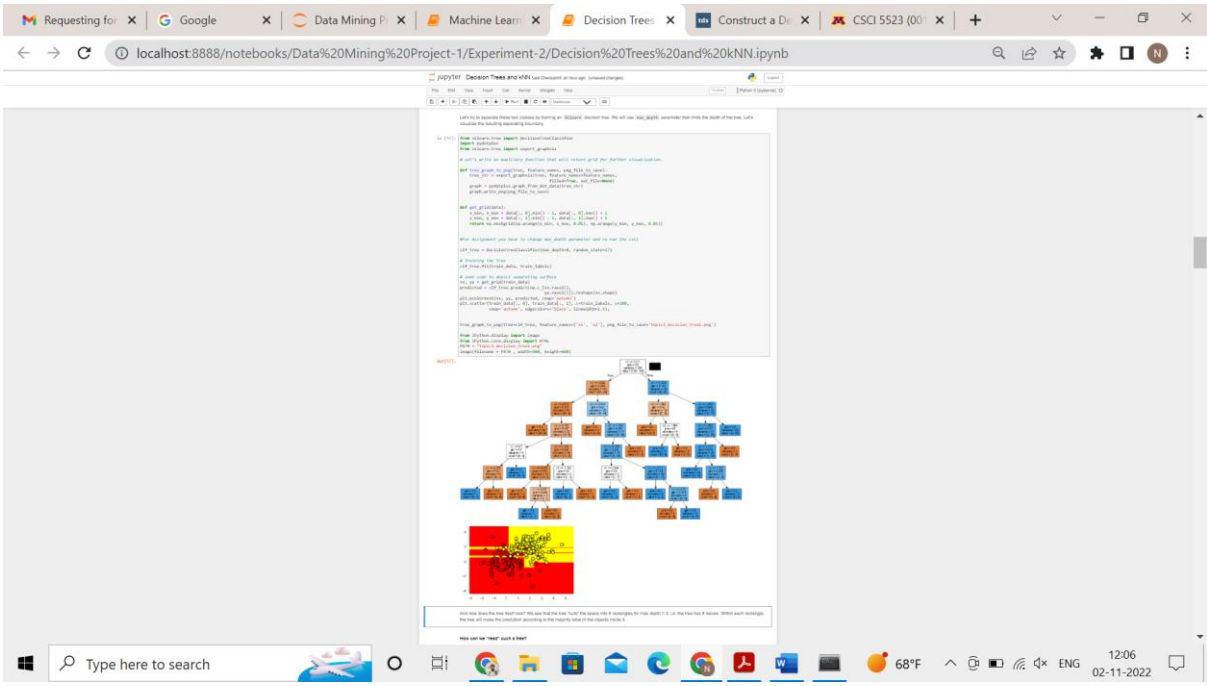


For Overfit (max_depth=7)

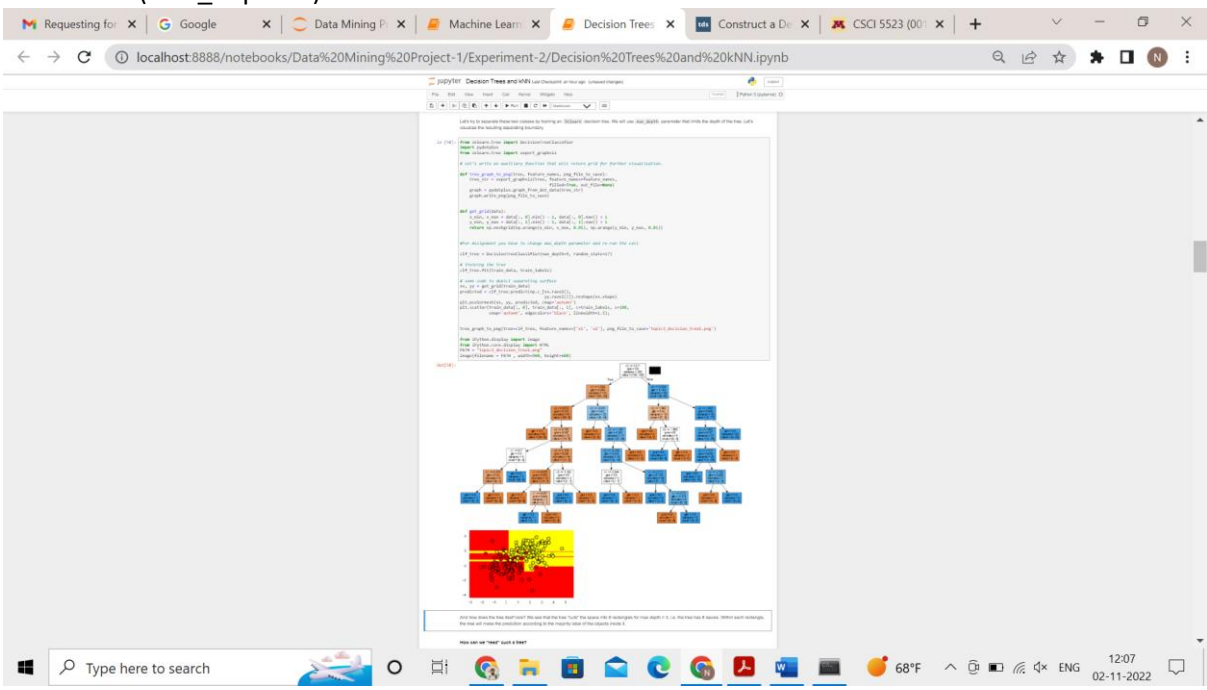


For Overfit (max_depth=8)

Project - 1

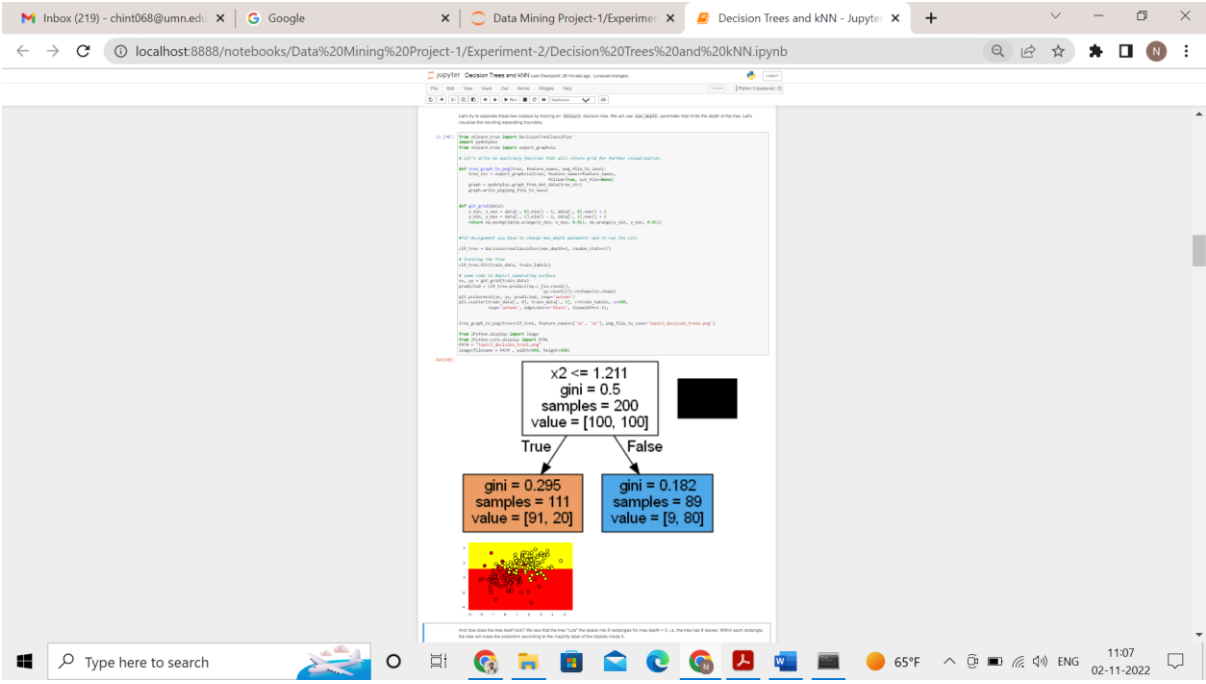


For Overfit (max_depth=9)

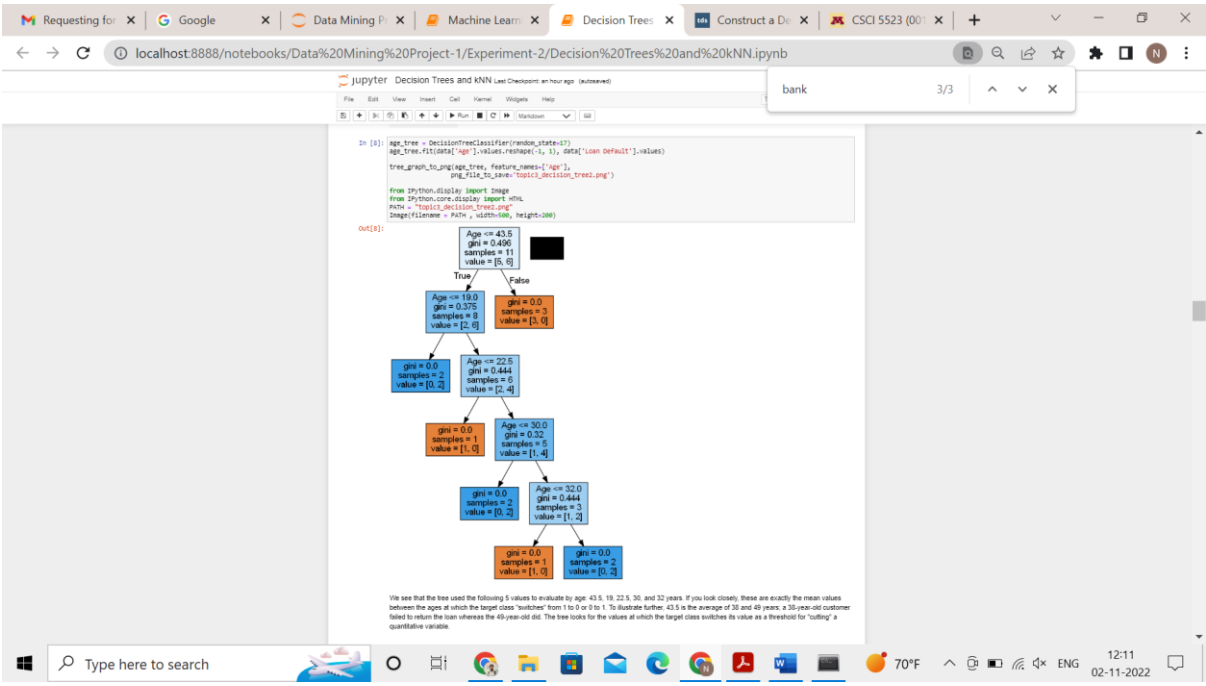


For Underfit (max_depth=1)

Project - 1



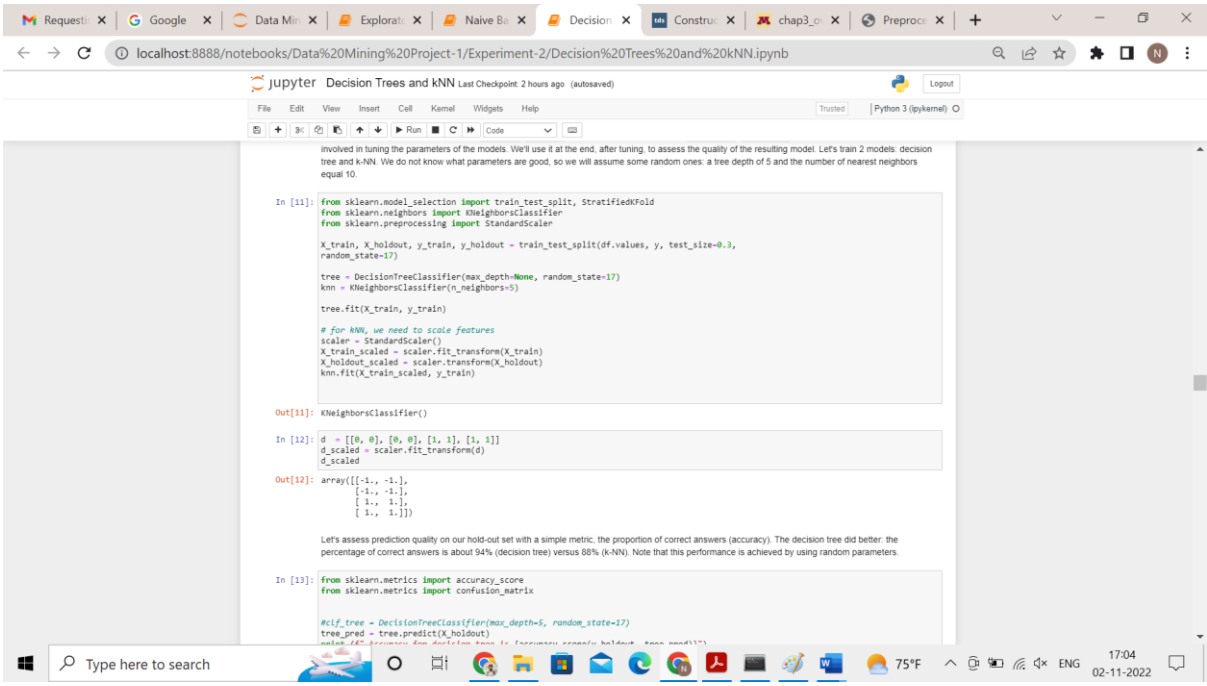
3.



We see that the tree used the following 5 values to evaluate by age: 43.5, 19, 22.5, 30, and 32 years. If you look closely, these are exactly the mean values between the ages at which the target class "switches" from 1 to 0 or 0 to 1.

4. Standard scalar standardizes the features by removing the mean and scaling to unit variance.

Project - 1



The scalar transformations are ([[-1., -1.], [-1., -1.], [1., 1.], [1., 1.]])

5. Small decision tree generally leads to under-fitting and big decision tree generally leads to overfitting. The training and test accuracy for overfitting decision tree model and under fitted decision tree model is performed against small and big data. Overfit model has high accuracy for training data but poor accuracy for the test data; in overfit model, test data accuracy is slightly better in the big data than small data because it has relatively high variance and low bias.
   Underfit model has poor accuracy for both training and test data irrespective of small data or big data because it high bias and low variance.

Project - 1

6.

```
==================================================
Balanced and Imbalanced dataset statistics
==================================================
For Balance Training Dataset, shape is (700, 18) and we have 350 positive class, and 350 negative class
For ImBalance Training Dataset, shape is (1993, 18) and we have 10 positive class, and 1983 negative class
For Balance Testing Dataset, shape is (266, 18) and we have 133 positive class, and 133 negative class
For ImBalance Testing Dataset, shape is (872, 18) and we have 5 positive class, and 867 negative class


==================================================
Train on balance and test on imbalance
==================================================
===============
[[350   0]
 [  0 350]]
===============
 Train Accuracy for decision tree trained on balanced data is 1.0
===============
[[688 179]
 [  0   5]]
===============
 while test accuracy on imbalanced data is 0.7947247706422018
 Train [f1_score, precision, recall] for decision tree trained on balanced data is (1.0, 1.0, 1.0)
 while [f1_score, precision, recall] on imbalanced data is (0.05291005291005291, 0.02717391304347826, 1.0)


==================================================
Train on balance and test on balance
==================================================
===============
[[350   0]
 [  0 350]]
===============
 Train Accuracy for decision tree trained on balanced data is 1.0
===============
[[108  25]
 [ 30 103]]
===============
 while test accuracy on balanced data is 0.793233082706767
 Train [f1_score, precision, recall] for decision tree trained on balanced data is (1.0, 1.0, 1.0)
 while test [f1_score, precision, recall] on balanced data is (0.7892720306513409, 0.8046875, 0.7744360902255639)


==================================================
Train on imbalance and test on balance
==================================================
===============
[[1983    0]
 [   0   10]]
===============
 Train Accuracy for decision tree trained on imbalanced data is 1.0
===============
[[133   0]
 [112  21]]
===============
 while test accuracy on balanced data is 0.578947368421052 7
 Train [f1_score, precision, recall] for decision tree trained on imbalanced data is (1.0, 1.0, 1.0)
 while test [f1_score, precision, recall] on balanced data is (0.2727272727272727, 1.0, 0.15789473684210525)


==================================================
Train on imbalance and test on imbalance
==================================================
===============
[[1983    0]
 [   0   10]]
===============
 Train Accuracy for decision tree trained on imbalanced data is 1.0
===============
[[864   3]
 [  5   0]]
===============
 while test accuracy on imbalanced data is  0.9908256880733946
 Train [f1_score, precision, recall] for decision tree trained on imbalanced data is (1.0, 1.0, 1.0)
 while test [f1_score, precision, recall] on imbalanced data is  (0.0, 0.0, 0.0)
```

First the statistics of the balanced and imbalanced data is seen. The model is trained on balanced dataset and tested on imbalanced dataset and same process is performed on other combinations too such as training on balanced data set and testing on balanced dataset; training on imbalanced dataset and testing on balanced dataset; training and testing on imbalanced dataset.

70% of the set is allocated to training and 30% for holdout. With this plit the model is trained and tested and a confusion matrix is created for each case and accordingly the corresponding F1 score, precision and recall are calculated using the in built sklearn functions.

	F1 score	**Precision**	**Recall**
Training balanced	1.0	1.0	1.0
Testing imbalanced	0.05291005291005291	, 0.02717391304347826	1.0
Training balanced	1.0	1.0	1.0
Testing balanced	0.7892720306513409	0.8046875	0.7744360902255639
Training imbalanced	1.0	1.0	1.0
Testing balanced	0.2727272727272727	1.0	0.15789473684210525
Training imbalanced	1.0	1.0	1.0

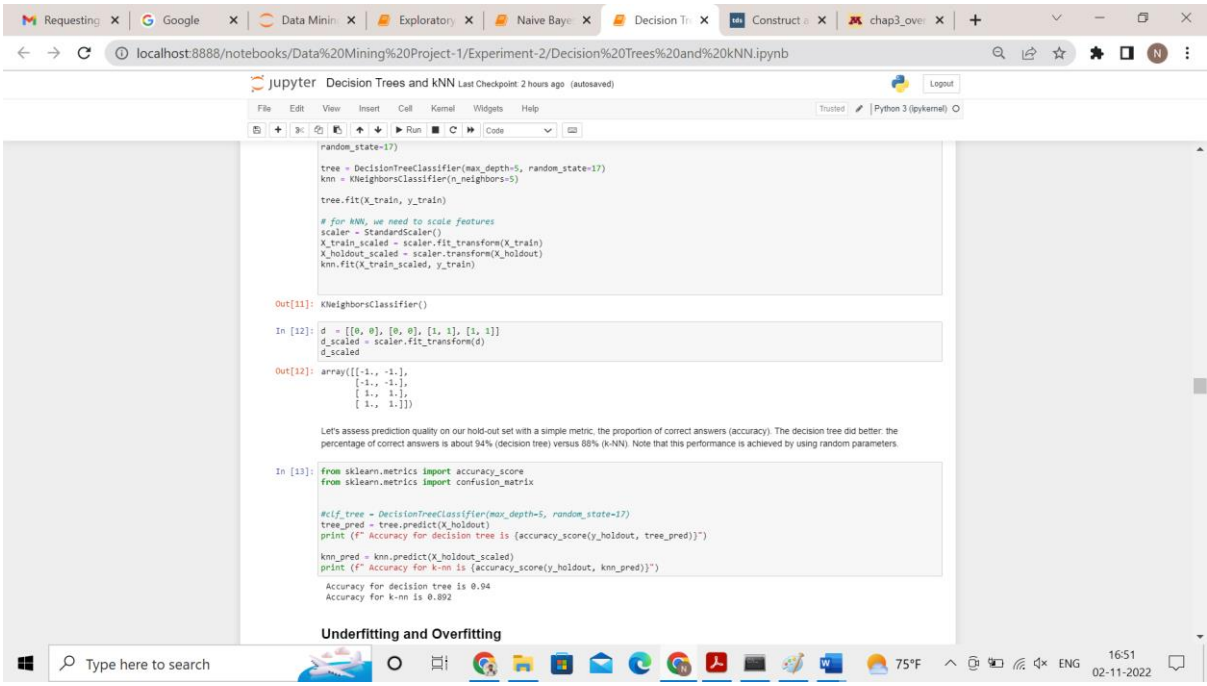
Project - 1

Testing imbalanced	0.0	0.0	0.0

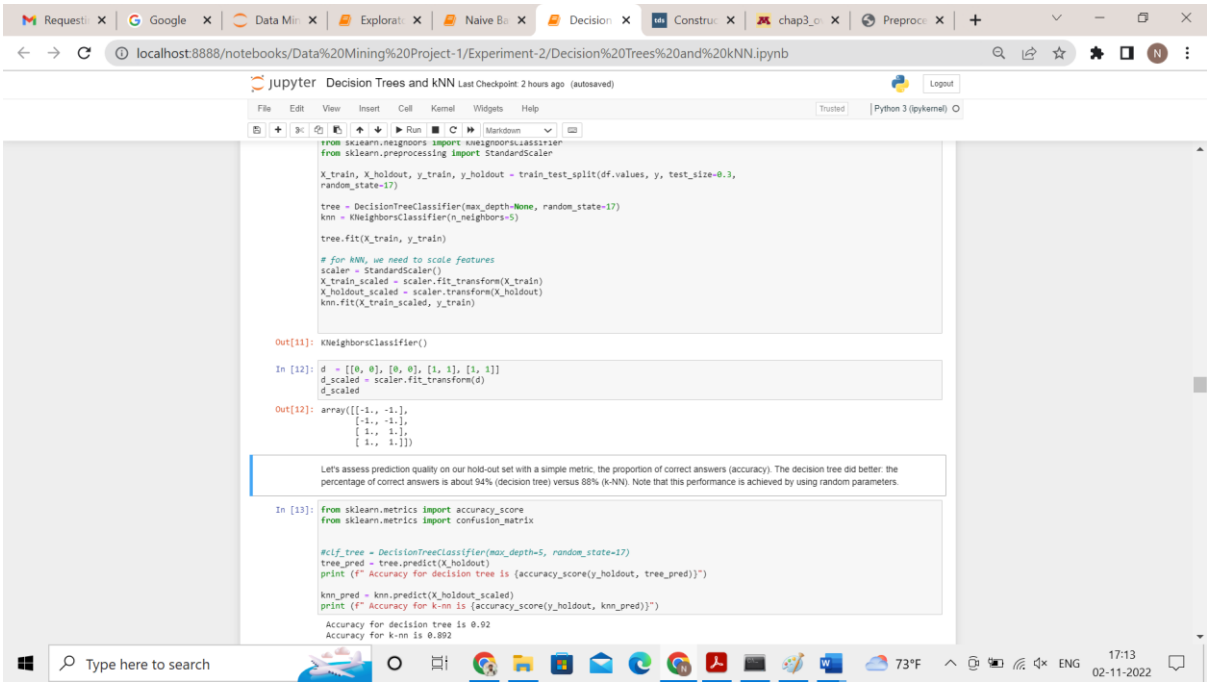
7.

In the experiment, the first the decision tree model is applied to the dataset and then some noise is added (irrelevant attributes) to both the test and training set and executed to check for the results. Adding irrelevant attributes reduces the accuracy of the model over the test dataset because of the model overfitting.

8.



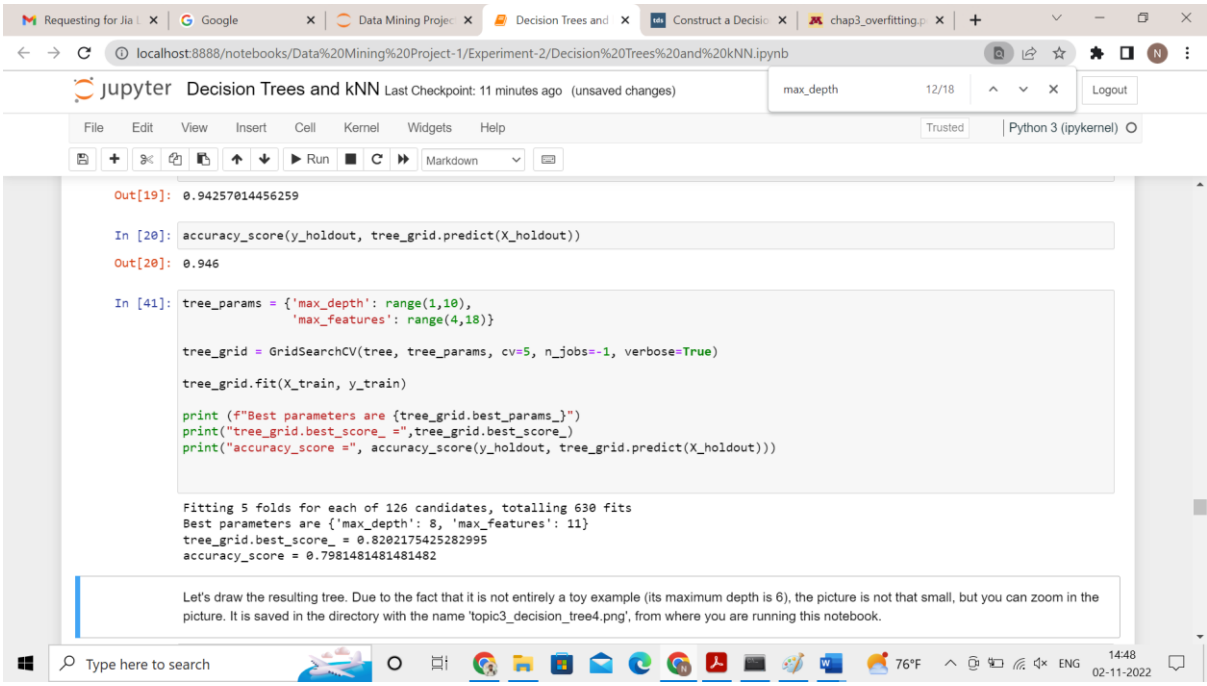
For max depth =5, the accuracy is 94%



For max depth is default, the accuracy is 92%. There is a decrease in the accuracy because of the overfitting of the model when max_depth is default value, when max_depth is default, the tree expands till the leaf nodes reach their pure state.

Project - 1

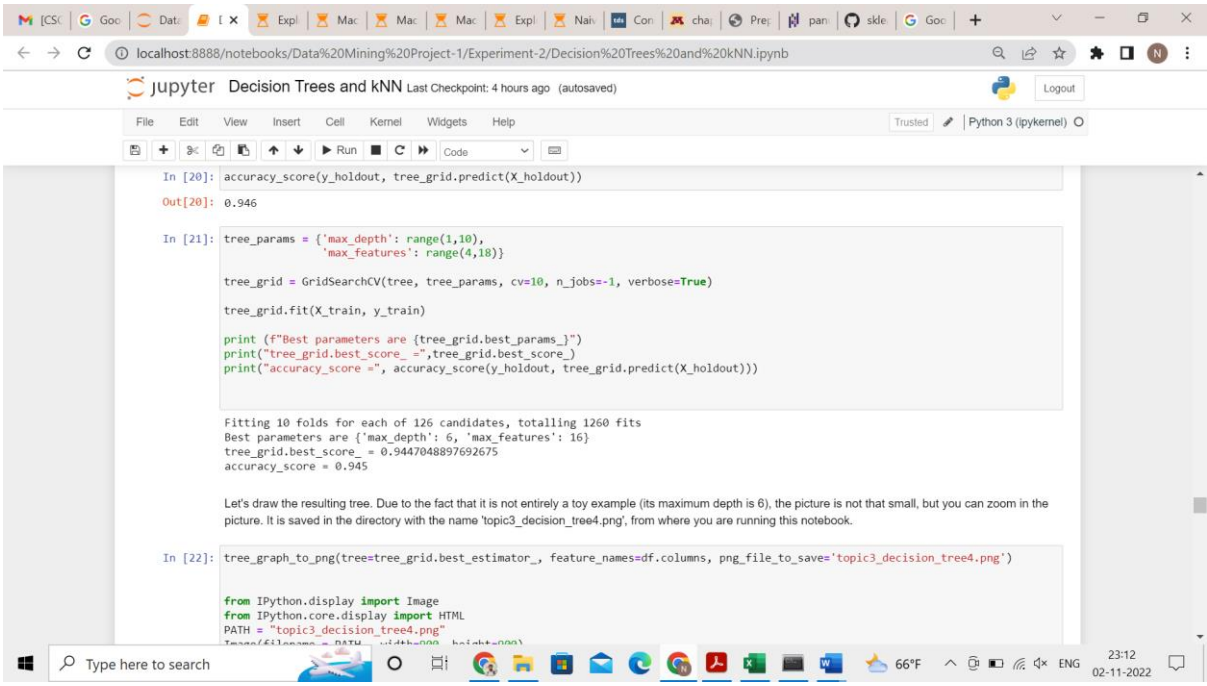
9.



```
Out[19]: 0.94257014456259

In [20]: accuracy_score(y_holdout, tree_grid.predict(X_holdout))
Out[20]: 0.946

In [41]: tree_params = {'max_depth': range(1,10),
                        'max_features': range(4,18)}

         tree_grid = GridSearchCV(tree, tree_params, cv=5, n_jobs=-1, verbose=True)

         tree_grid.fit(X_train, y_train)

         print (f"Best parameters are {tree_grid.best_params_}")
         print("tree_grid.best_score_ =",tree_grid.best_score_)
         print("accuracy_score =", accuracy_score(y_holdout, tree_grid.predict(X_holdout)))

         Fitting 5 folds for each of 126 candidates, totalling 630 fits
         Best parameters are {'max_depth': 8, 'max_features': 11}
         tree_grid.best_score_ = 0.8202175425282995
         accuracy_score = 0.7981481481481482
```

Let's draw the resulting tree. Due to the fact that it is not entirely a toy example (its maximum depth is 6), the picture is not that small, but you can zoom in the picture. It is saved in the directory with the name 'topic3_decision_tree4.png', from where you are running this notebook.

A total of 630 decision trees can be constructed



```
In [20]: accuracy_score(y_holdout, tree_grid.predict(X_holdout))
Out[20]: 0.946

In [21]: tree_params = {'max_depth': range(1,10),
                        'max_features': range(4,18)}

         tree_grid = GridSearchCV(tree, tree_params, cv=10, n_jobs=-1, verbose=True)

         tree_grid.fit(X_train, y_train)

         print (f"Best parameters are {tree_grid.best_params_}")
         print("tree_grid.best_score_ =",tree_grid.best_score_)
         print("accuracy_score =", accuracy_score(y_holdout, tree_grid.predict(X_holdout)))

         Fitting 10 folds for each of 126 candidates, totalling 1260 fits
         Best parameters are {'max_depth': 6, 'max_features': 16}
         tree_grid.best_score_ = 0.9447048897692675
         accuracy_score = 0.945
```

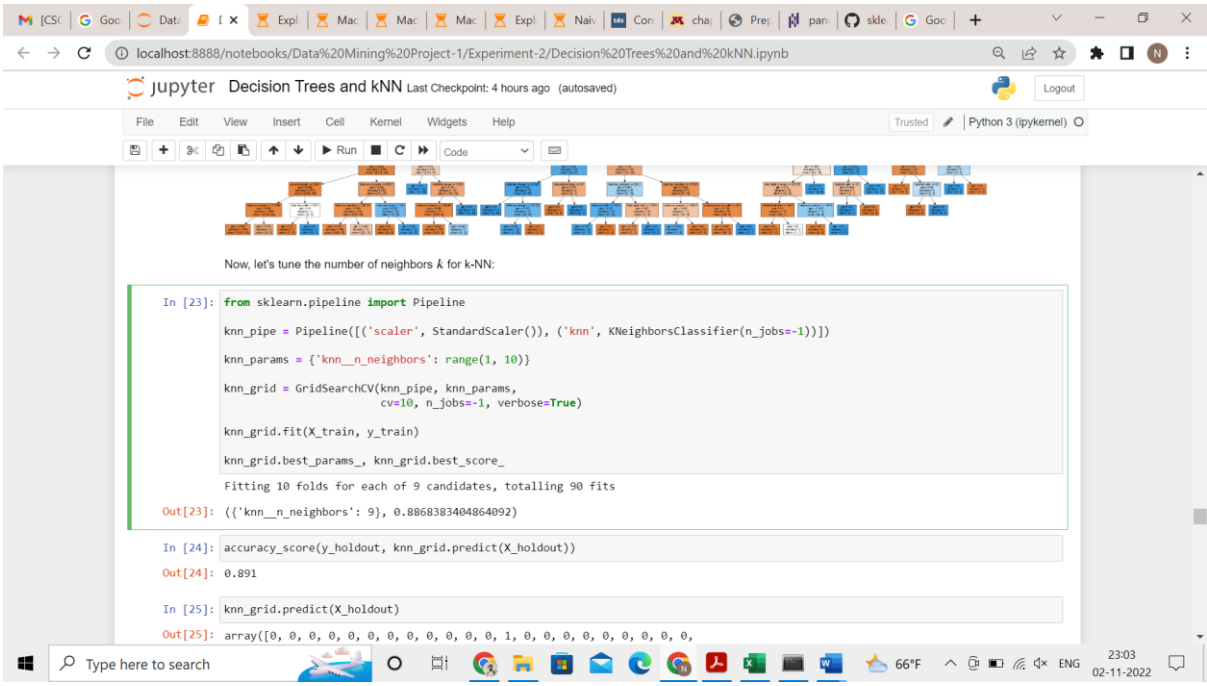
Let's draw the resulting tree. Due to the fact that it is not entirely a toy example (its maximum depth is 6), the picture is not that small, but you can zoom in the picture. It is saved in the directory with the name 'topic3_decision_tree4.png', from where you are running this notebook.

```
In [22]: tree_graph_to_png(tree=tree_grid.best_estimator_, feature_names=df.columns, png_file_to_save='topic3_decision_tree4.png')

         from IPython.display import Image
         from IPython.core.display import HTML
         PATH = "topic3_decision_tree4.png"
```

A total of 1260 decision trees can be constructed.

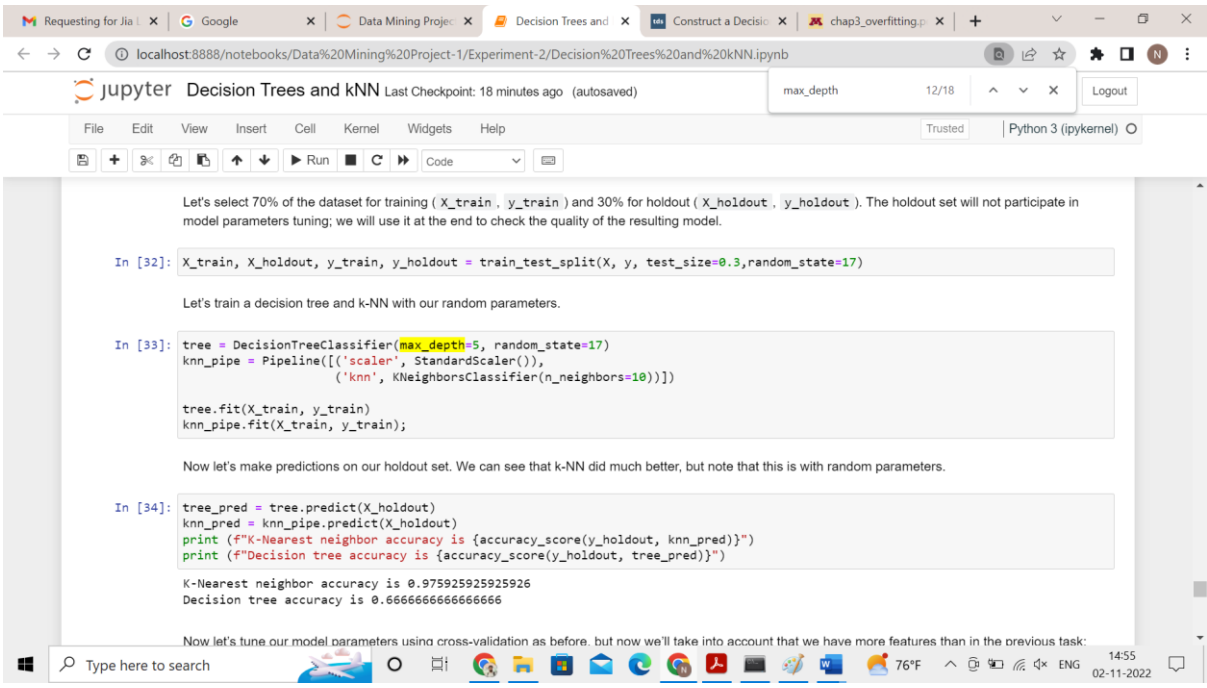
Project - 1

10.



k=9 is the best choice

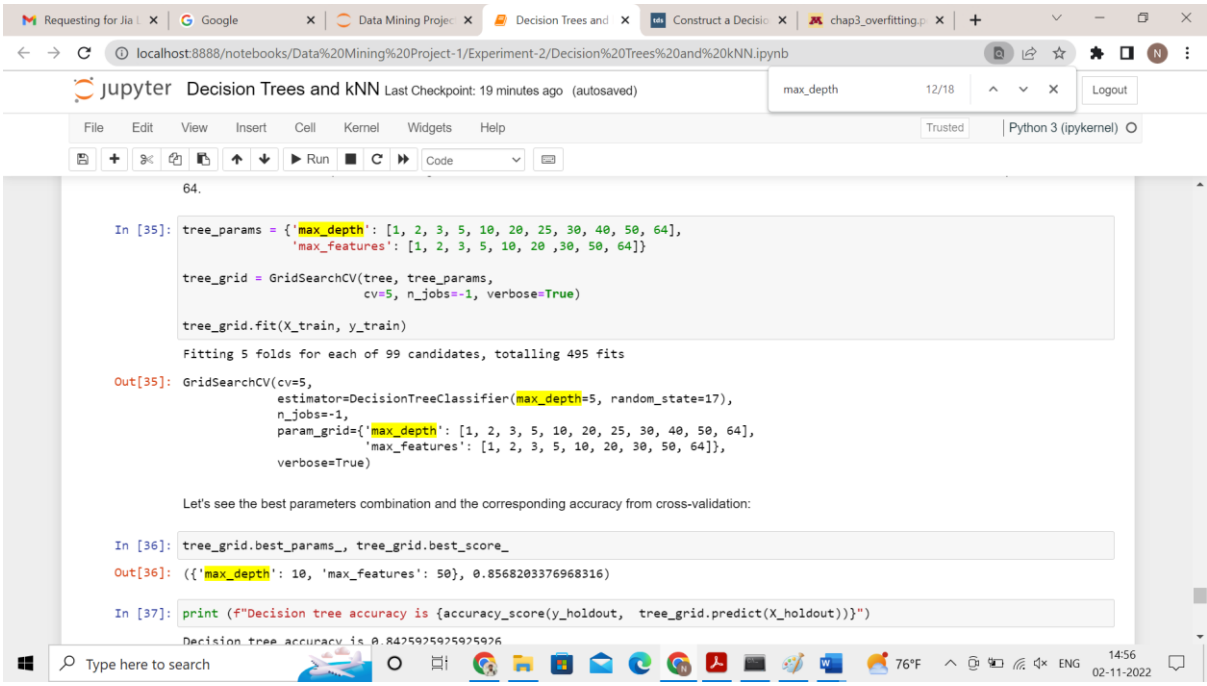
11.



Accuracy of the decision tree for max depth = 5 is 66.67%

Accuracy of the kNN when K=10 is 97.59%

Project - 1



```
In [35]: tree_params = {'max_depth': [1, 2, 3, 5, 10, 20, 25, 30, 40, 50, 64],
                        'max_features': [1, 2, 3, 5, 10, 20 ,30, 50, 64]}

         tree_grid = GridSearchCV(tree, tree_params,
                                  cv=5, n_jobs=-1, verbose=True)

         tree_grid.fit(X_train, y_train)

         Fitting 5 folds for each of 99 candidates, totalling 495 fits

Out[35]: GridSearchCV(cv=5,
                      estimator=DecisionTreeClassifier(max_depth=5, random_state=17),
                      n_jobs=-1,
                      param_grid={'max_depth': [1, 2, 3, 5, 10, 20, 25, 30, 40, 50, 64],
                                  'max_features': [1, 2, 3, 5, 10, 20, 30, 50, 64]},
                      verbose=True)
```

Let's see the best parameters combination and the corresponding accuracy from cross-validation:

```
In [36]: tree_grid.best_params_, tree_grid.best_score_

Out[36]: ({'max_depth': 10, 'max_features': 50}, 0.8568203376968316)

In [37]: print (f"Decision tree accuracy is {accuracy_score(y_holdout,  tree_grid.predict(X_holdout))}")

         Decision tree accuracy is 0.8425925925925926
```

With 5-fold cross validation, best parameters are max_depth = 10 and max_features=50 and accuracy of holdout dataset for the decision tree = 85.68%