DATA MINING- CLASSIFICATION PROBLEMS

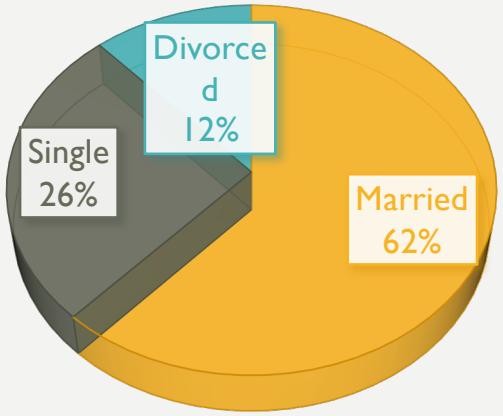
SUBMITTED BY:
SEERAT CHHABRA

DIRECT MARKETING

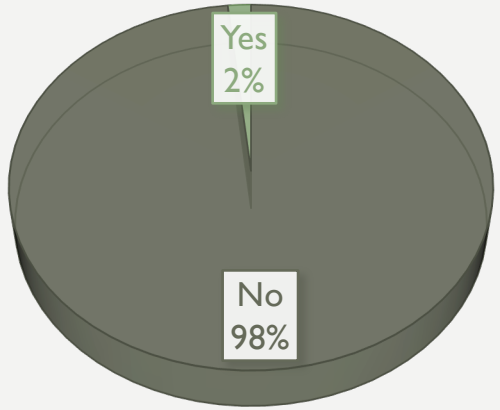
PORTUGUESE BANK

DATA SUMMARY (1)

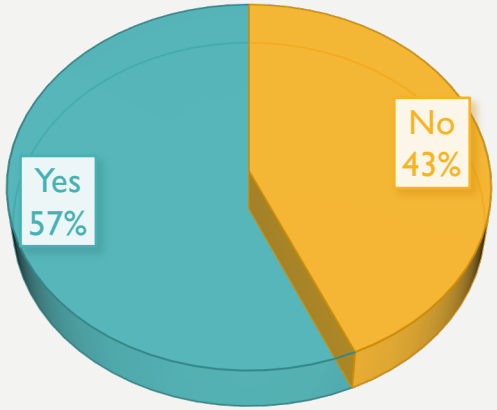
MARITAL STATUS



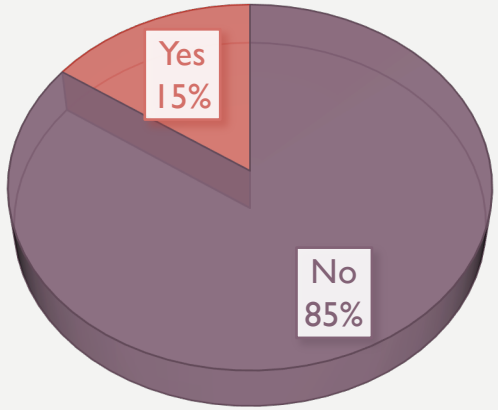
DEFAULT



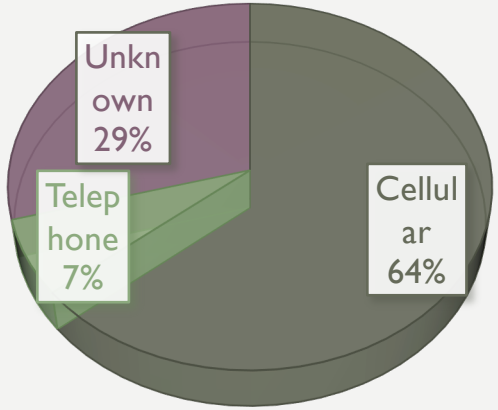
HOUSING LOAN



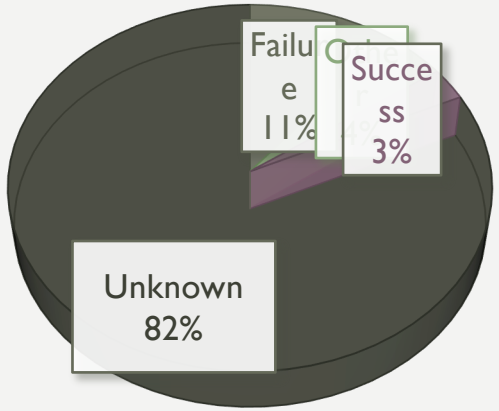
PERSONAL LOAN



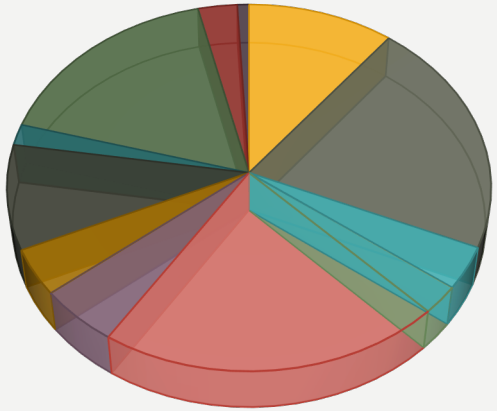
CONTACT



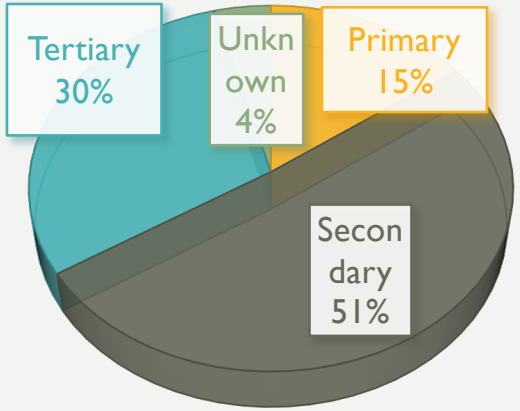
PREVIOUS OUTCOME



JOB

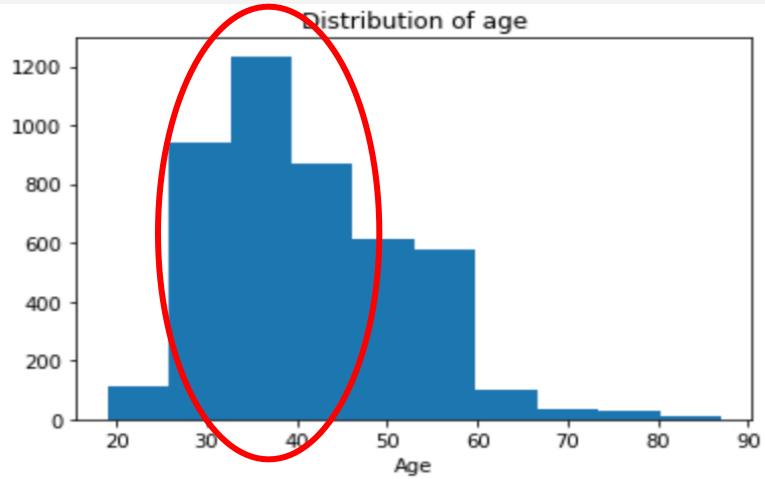


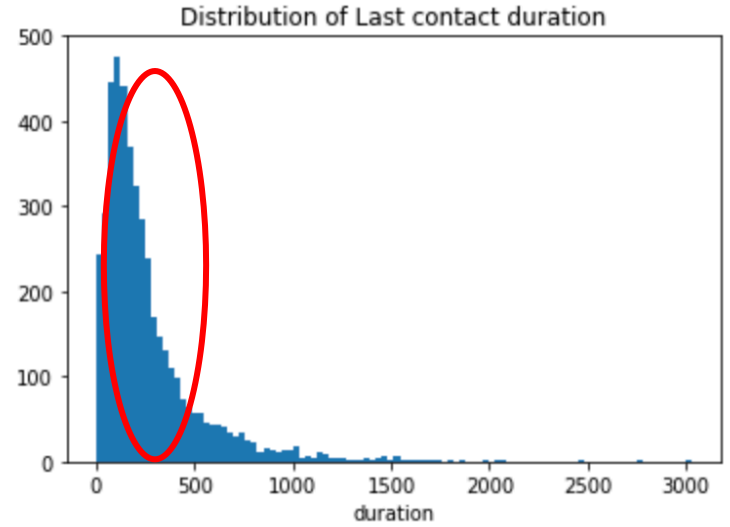
EDUCATION



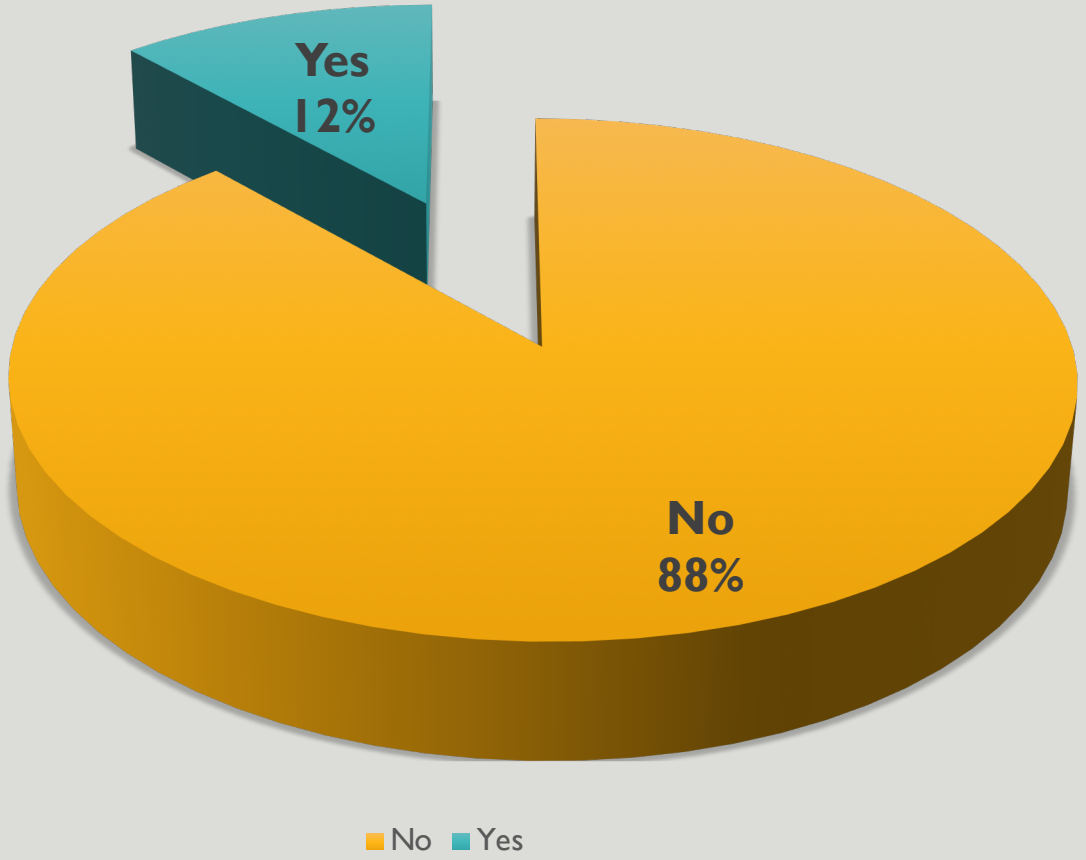
DATA SUMMARY (2)

	age	balance	duration	campaign	pdays	previous
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	41.170095	1422.657819	263.961292	2.793630	39.766645	0.542579
std	10.576211	3009.638142	259.856633	3.109807	100.121124	1.693562
min	19.000000	-3313.000000	4.000000	1.000000	-1.000000	0.000000
25%	33.000000	69.000000	104.000000	1.000000	-1.000000	0.000000
50%	39.000000	444.000000	185.000000	2.000000	-1.000000	0.000000
75%	49.000000	1480.000000	329.000000	3.000000	-1.000000	0.000000
max	87.000000	71188.000000	3025.000000	50.000000	871.000000	25.000000



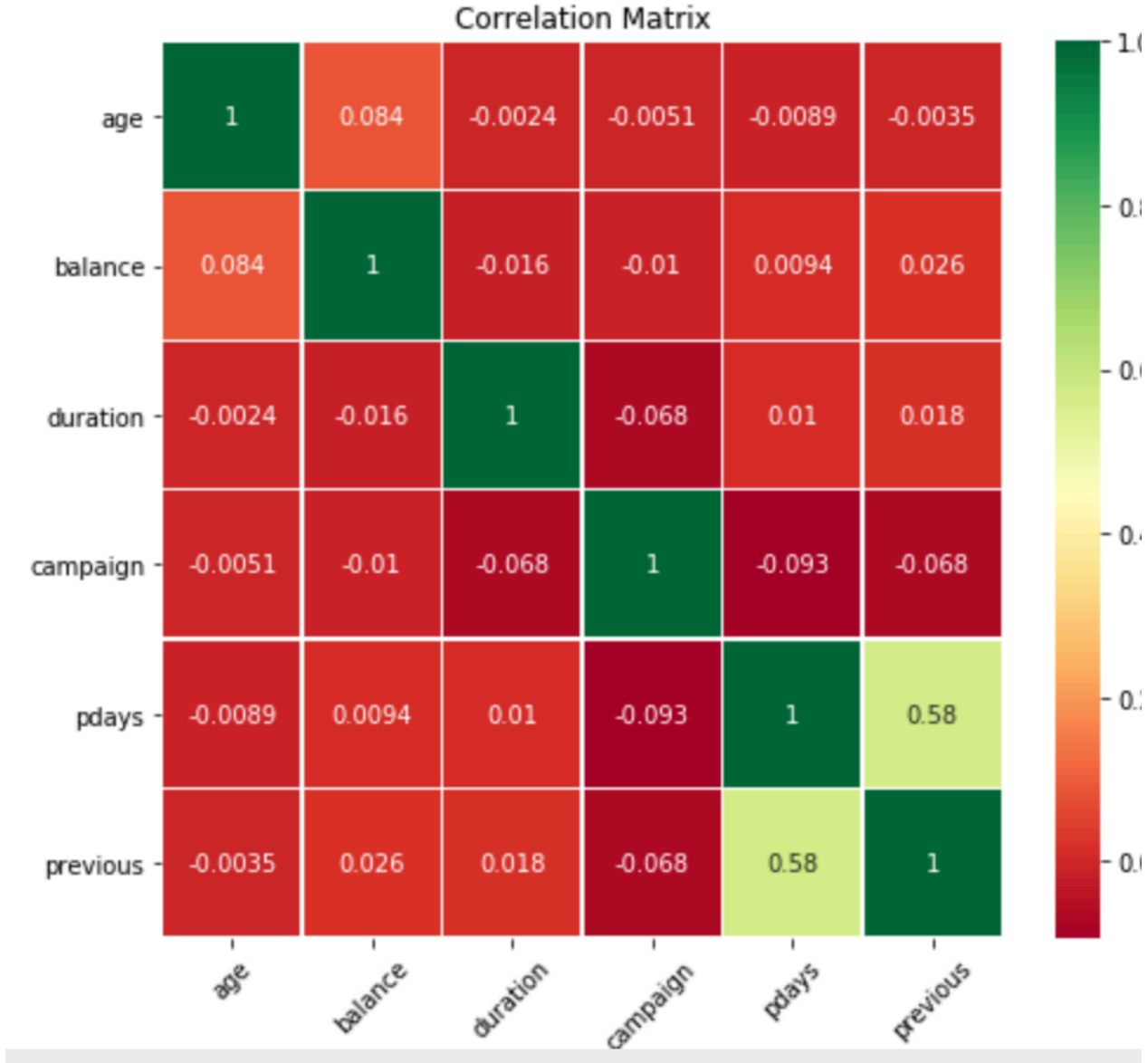


# of Clients who subscribed to term deposit



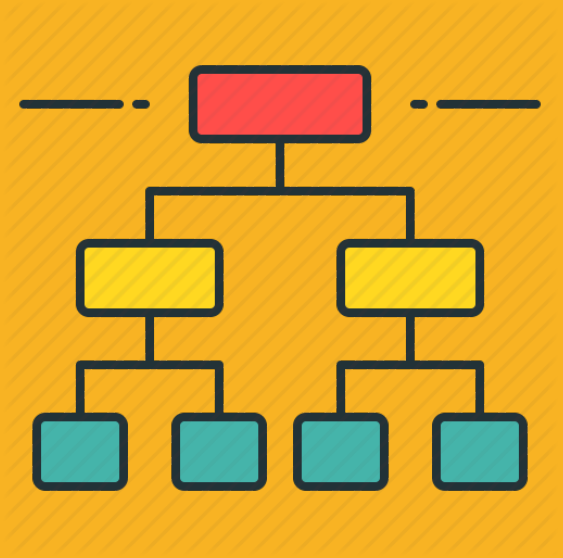
UNBALANCED DATASET

NO HIGHLY CORELATED ATTRIBUTES!



DATA PRE-PROCESSING

- Removed attribute day which captures last contact day of month
- More relevant attribute is number of days since last contact



DECISION TREE ALGORITHM

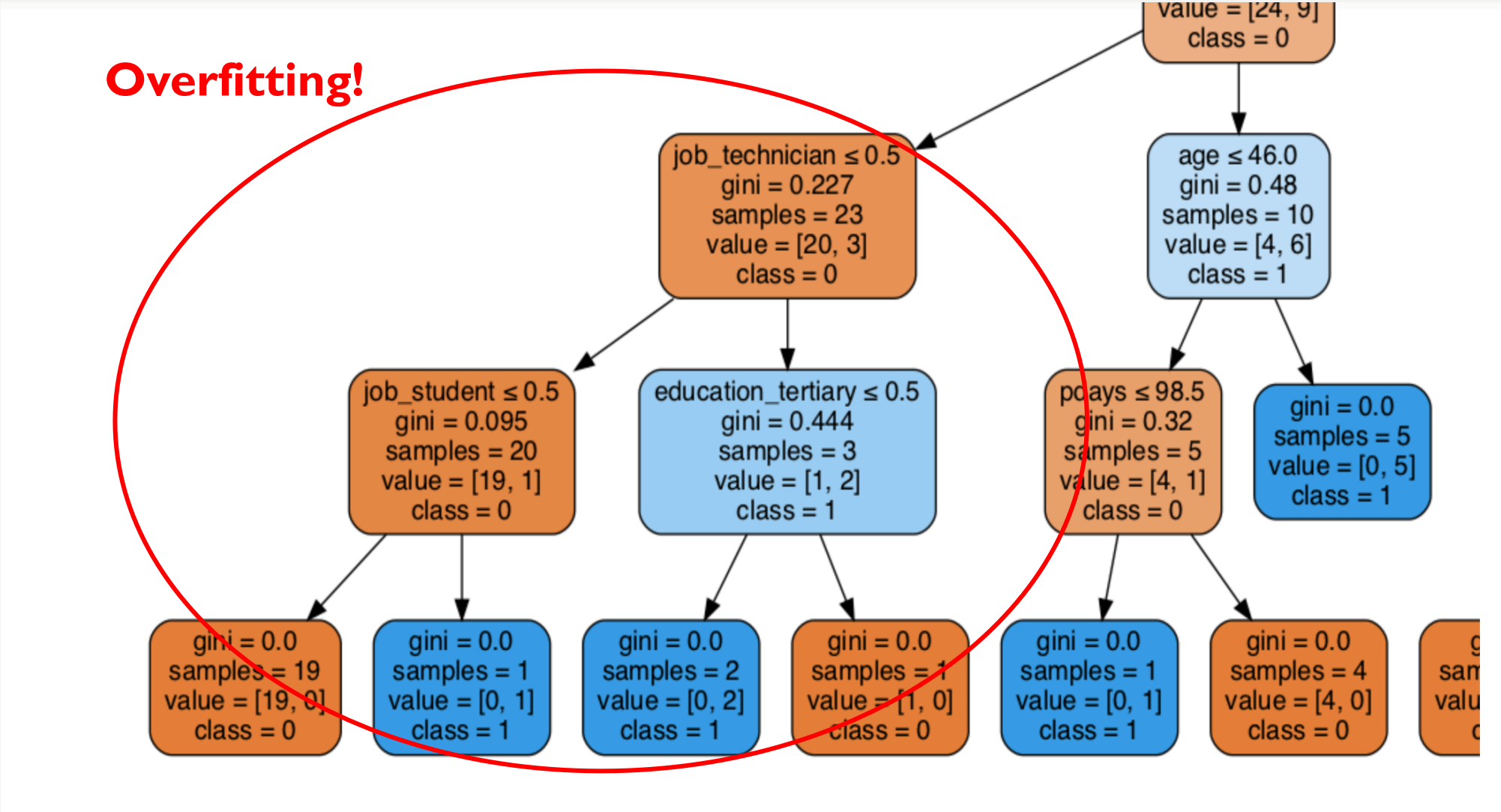
DEFAULT DECISION TREE MODEL

```
#Make default decision tree using training dataset
clf = DecisionTreeClassifier()
clf.fit(X_train, Y_train)

#Prediction on training data
pred=pd.DataFrame(clf.predict(X_train),columns=["Prediction"])
print("--------------------------Decision Tree - Training Data-------------------
print("Accuracy Score on training data using Decision Tree:",accuracy_score(Y_train,
print("Confusion Matrix on training data using Decision Tree\n", confusion_matrix(Y_

# prediction on test data
pred=pd.DataFrame(clf.predict(X_test),columns=["Prediction"])
print("--------------------------Decision Tree - Test Data-----------------------------------------------\n")
print("Accuracy Score on test data using Decision Tree:",accuracy_score(Y_test,pred["Prediction"]))
print("Balanced Accuracy Score on test data using Decision Tree:",balanced_accuracy_score(Y_test,pred["Prediction"]))
print("Confusion Matrix on test data using Decision Tree\n", confusion_matrix(Y_test,pred["Prediction"]))
print("Classification report\n", classification_report(Y_test,pred["Prediction"]))
```

Default Parameters:

- Criterion: Gini
- Max depth of tree: None
- Min samples in leaf node: 1
- Min samples in node to split: 2

PORTION OF DEFAULT DECISION TREE



CONFUSION MATRIX ON TEST DATA

	Predicted won't purchase	Predicted will purchase
Actual won't purchase	37087	2835
Actual will purchase	2584	2705

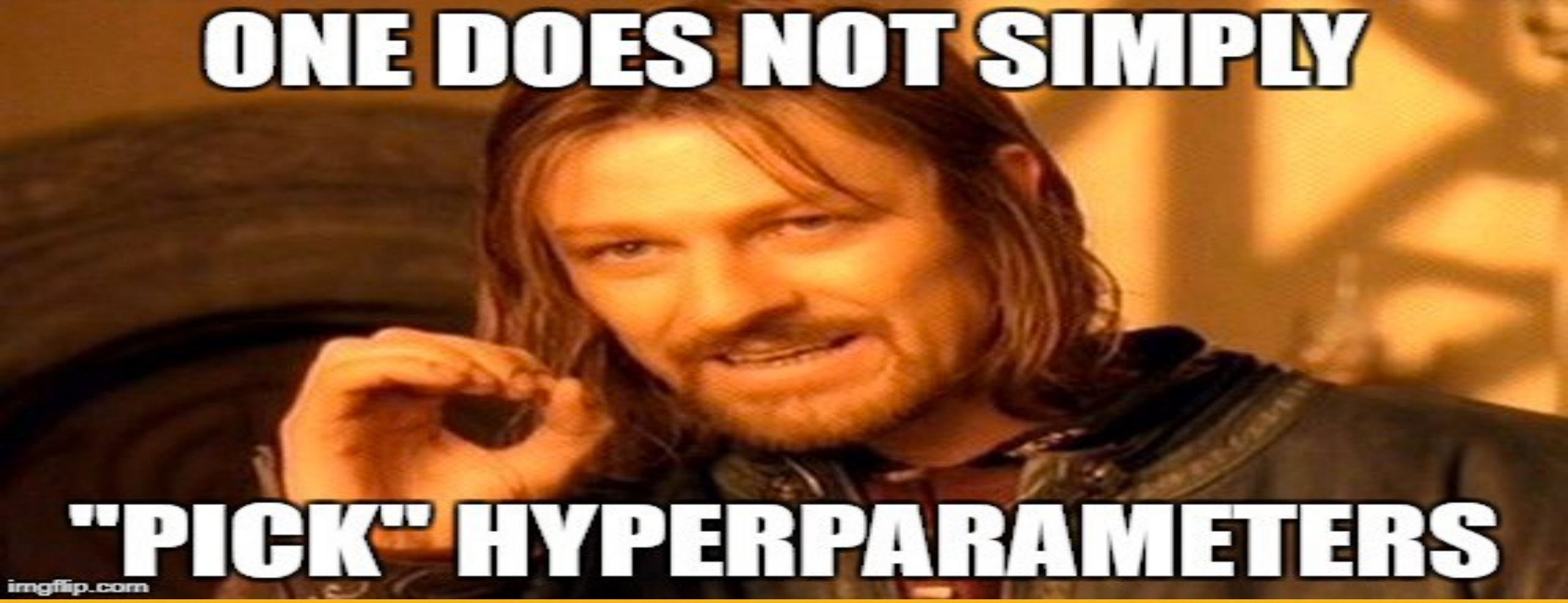
True Positive

False Positive

False Negative

True Negative

Precision of who will purchase ($\frac{2705}{(2705+2835)}$) = 49%



HYPERPARAMETER TUNING

RANDOM AND GRID SEARCH

```
# For this particular example, i want my precision to be more for who will buy..
#I want to correctly predict more people who will actually buy
#Hyperparameter tuning done for decision tree classifier
#RANDOM SEARCH-----------------------------------------

print("RandomizedSearchCV-Decision tree")
parameters={'max_depth': range(10,100,10), 'min_samples_leaf' : range(2,30,5), 'criterion':['gini','entropy']}
clf_random = RandomizedSearchCV(clf,parameters,n_iter=30, cv=8, scoring ="precision")
clf_random.fit(X_train, Y_train)
rand_parm=clf_random.best_params_
print(rand_parm)

#GRID SEARCH--------------------------------------
print("GridSearchCV-Decision tree")
clf_grid = GridSearchCV(clf,parameters, scoring ="precision")
clf_grid.fit(X_train, Y_train)
grid_parm=clf_grid.best_params_
print(grid_parm)
```

Parameters for variation:

- Max depth of tree: (10,100,10)
- Min samples in the leaf node: (2,30,5)
- Criterion for splitting: (Gini, Entropy)

Optimize Precision score!

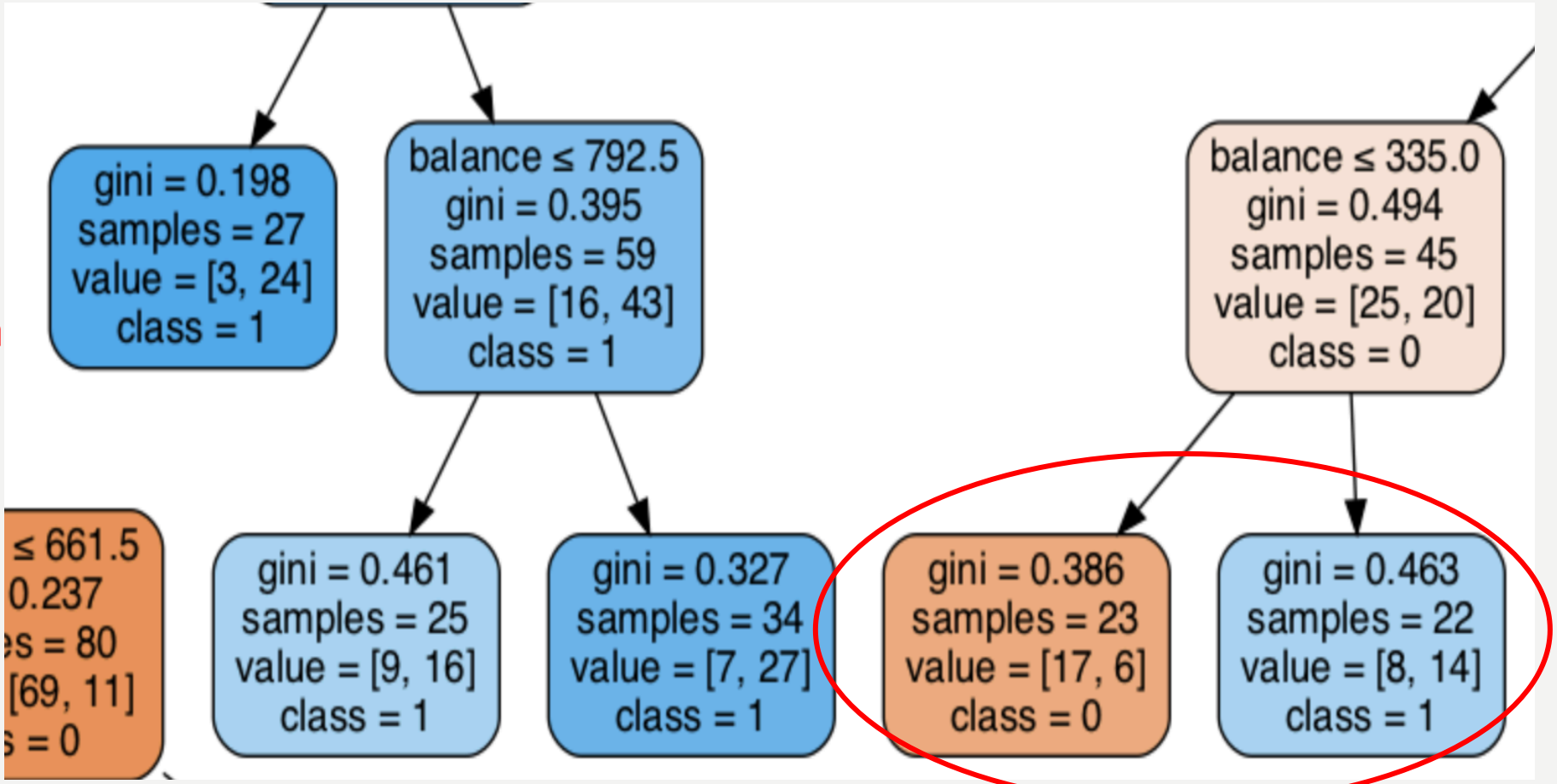
Want to correctly predict more customers who will actually buy deposits!

```
RandomizedSearchCV-Decision tree
{'min_samples_leaf': 12, 'max_depth': 10, 'criterion': 'entropy'}
GridSearchCV-Decision tree
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 22}
```

Optimal Parameters

DECISION TREE USING GRID SEARCH PARAMETERS

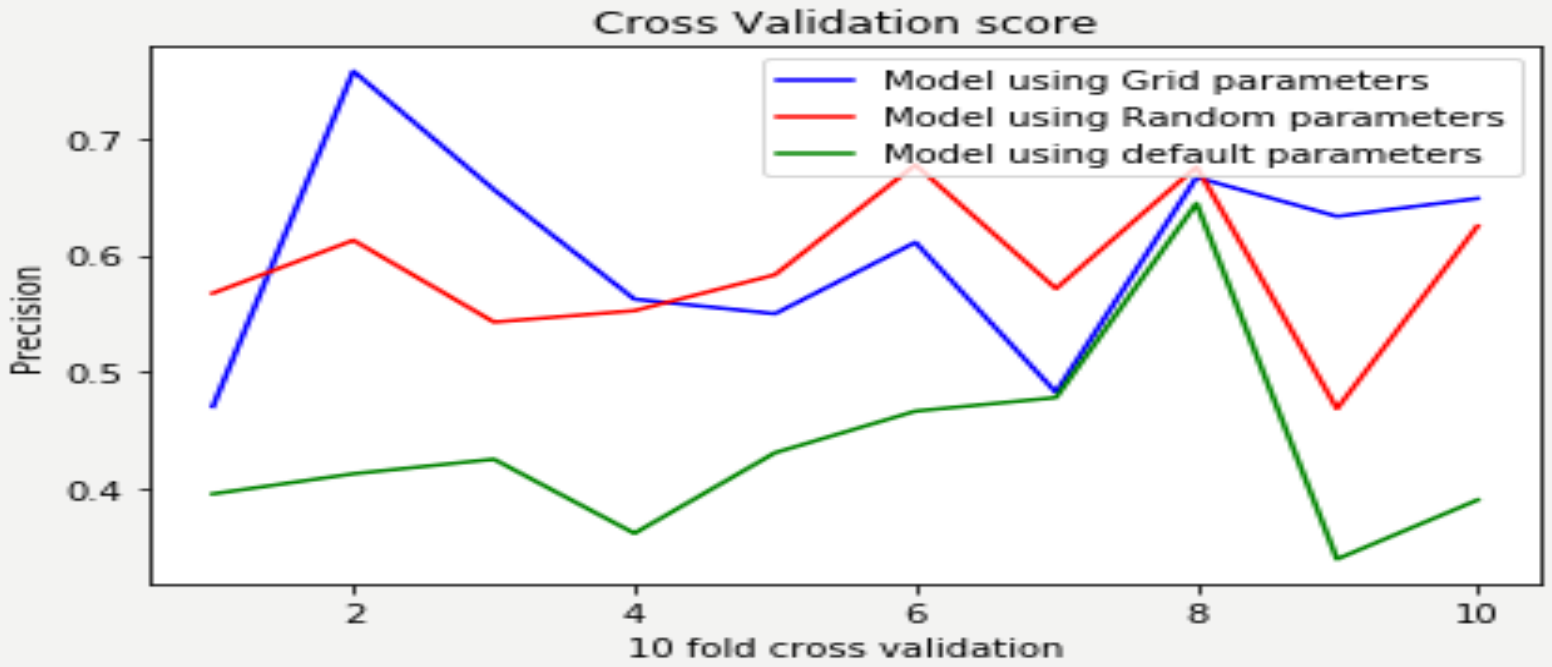
Parameter Min samples in leaf set to 22 prevents overfitting



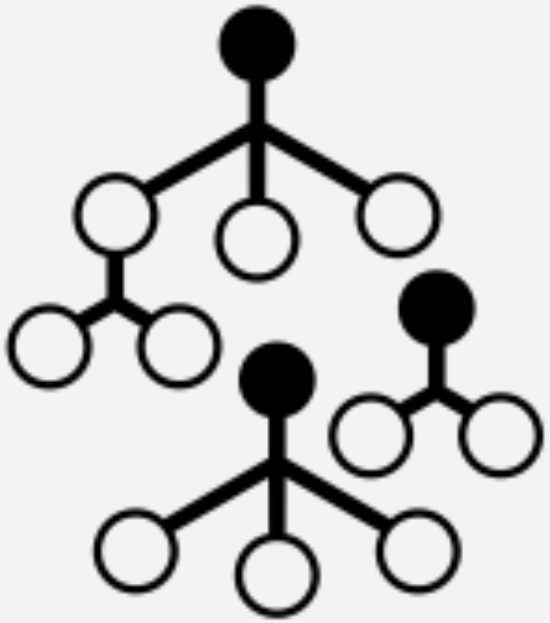
MODEL PERFORMANCE ON TEST DATA

	Default model	Grid Search model	Random Search model
Accuracy	88.01%	89.97%	89.92%
Balanced Accuracy	72.02%	65.63%	68.35%
Precision of who purchased	49%	**63%**	60%
Recall of who purchased	51%	34%	40%
F1 – score of who purchased	50%	44%	48%

CROSS VALIDATION



	Default model	Grid Search model	Random Search model
Mean Precision score of cross validation	43.47%	**60.39%**	58.77%



RANDOM FOREST ALGORITHM

DEFAULT RANDOM FOREST MODEL

```
#Random forest
rfc = RandomForestClassifier()
rfc.fit(X_train, Y_train)

#Prediction on training data
pred=pd.DataFrame(rfc.predict(X_train),columns=["Prediction"])
print("--------------------------Random Forest: Training Data----------------------
print("Accuracy Score on training data using Random Forest:",accuracy_score(Y_train,pr
print("Confusion Matrix on training data using Random Forest\n", confusion_matrix(Y_tr

# prediction on test data
pred=pd.DataFrame(rfc.predict(X_test),columns=["Prediction"])
print("--------------------------Random Forest: Test Data-----------------------------------------------\n")
print("Accuracy Score on test data using Random Forest:",accuracy_score(Y_test,pred["Prediction"]))
print("Confusion Matrix on test data using Random Forest\n", confusion_matrix(Y_test,pred["Prediction"]))
print("Classification report on test data using Random Forest\n", classification_report(Y_test,pred["Prediction"]))
```

Default Parameters:

- Max features: Auto
- Number of tress: 10
- Criterion: Gini
- Max depth of tree: None
- Min samples in leaf node: 1
- Min samples in node to split: 2

HYPERPARAMETER TUNING

RANDOM AND GRID SEARCH

```
#Hyperparameter tuning done for random forest classifier
#RANDOM SEARCH----------------------------------------

print("RandomizedSearchCV-Decision tree")
parameters={'max_features': range(10,50,5) ,'n_estimators':[25,30,40,50], 'criterion':['gini','entropy']}
rfc_random = RandomizedSearchCV(rfc,parameters,n_iter=30,cv=5, scoring ='precision')
rfc_random.fit(X_train, Y_train)
rand_parm_rfc=rfc_random.best_params_
print(rand_parm_rfc)

#GRID SEARCH-------------------------------------
print("GridSearchCV-Decision tree")
rfc_grid = GridSearchCV(rfc,parameters,scoring ='precision')
rfc_grid.fit(X_train, Y_train)
grid_parm_rfc=rfc_grid.best_params_
print(grid_parm_rfc)
```

Parameters for variation:

- Max features: (10,50,5)
- Number of trees: (25,30,40,50)
- Criterion for splitting: (Gini, Entropy)

Optimize Precision score!

Want to correctly predict more customers who will actually buy!

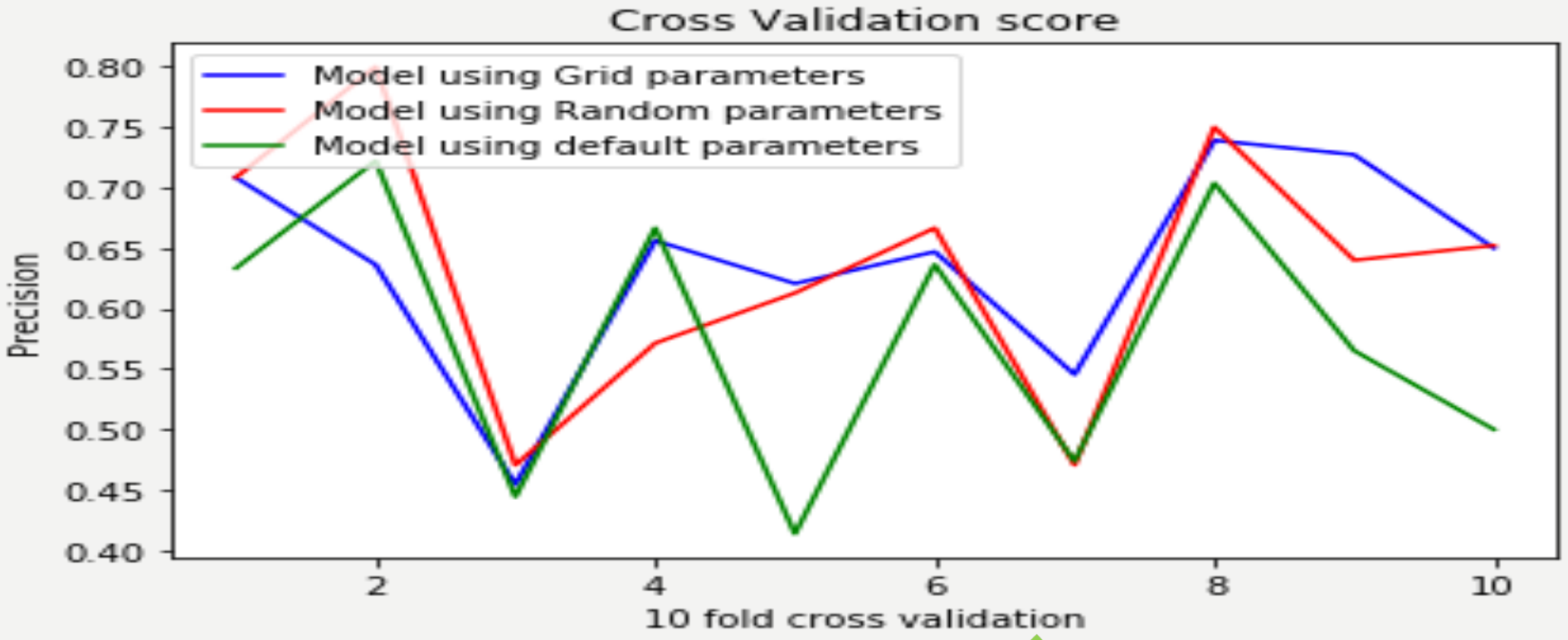
```
RandomizedSearchCV-Decision tree
{'n_estimators': 40, 'max_features': 10, 'criterion': 'entropy'}
GridSearchCV-Decision tree
{'criterion': 'entropy', 'max_features': 10, 'n_estimators': 50}
```

Optimal Parameters

MODEL PERFORMANCE ON TEST DATA

	Default model	Grid Search model	Random Search model
Accuracy	90.43%	91.04%	91.12%
Balanced Accuracy	65.32%	68.60%	69.18%
Precision of who purchased	69%	**71%**	**71%**
Recall of who purchased	33%	39%	41%
F1 – score of who purchased	44%	51%	52%

CROSS VALIDATION



	Default model	Grid Search model	Random Search model
Mean Precision score of cross validation	57.59%	**63.85%**	63.42%

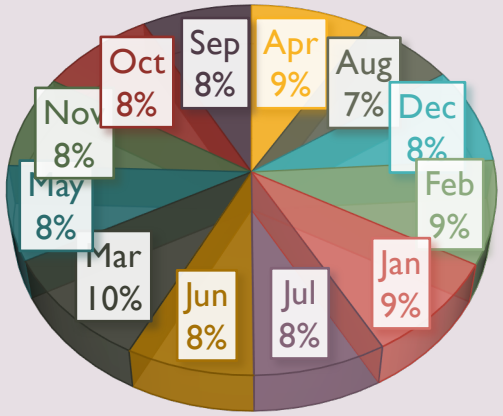
Overall Random Forest with parameters from grid search is working best on test data with maximum precision – 71%!

INSURANCE FRAUD DETECTION

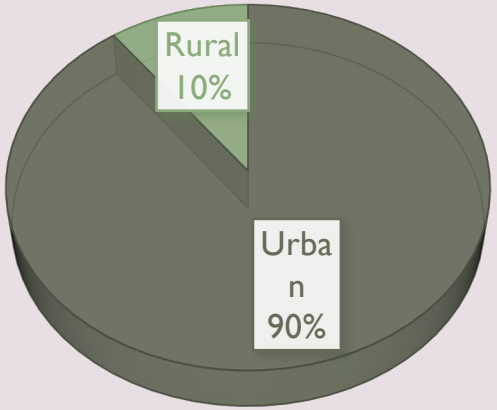
AUTO INSURANCE COMPANY

DATA SUMMARY (1)

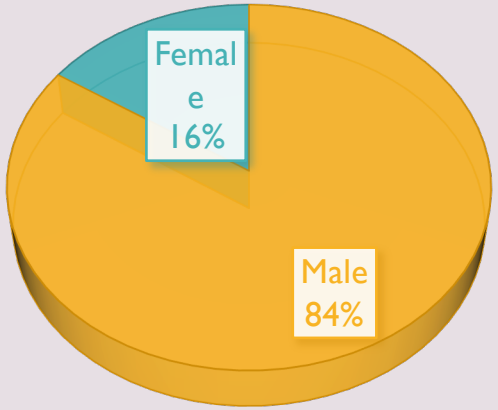
ACCIDENT MONTH



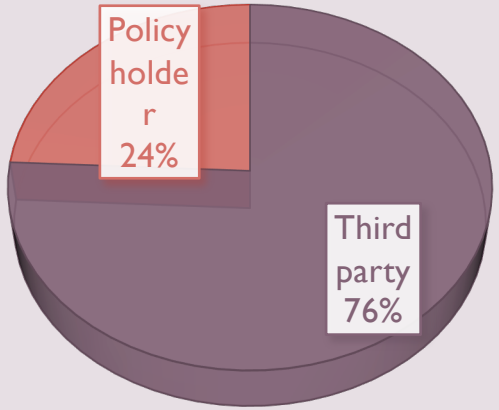
ACCIDENT AREA



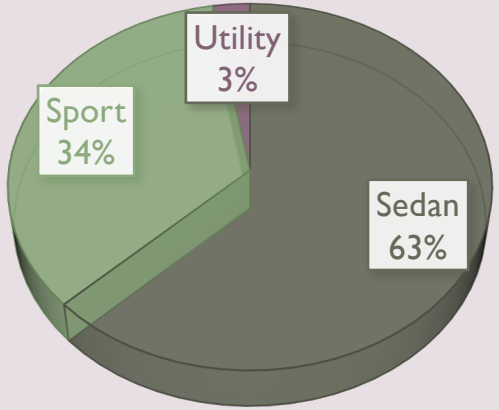
SEX



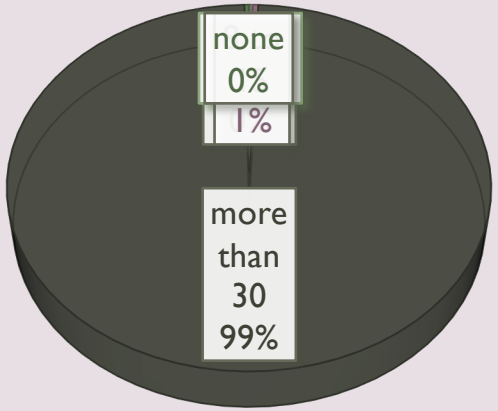
FAULT



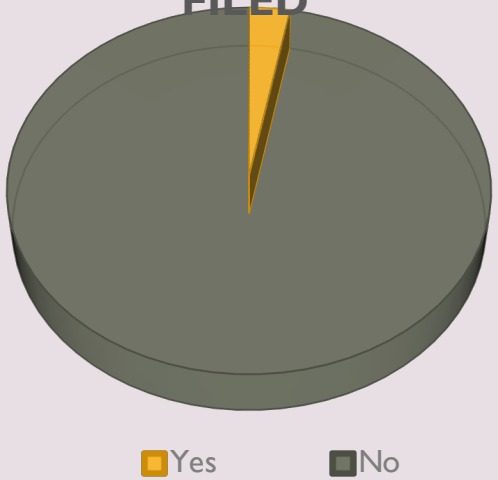
VEHICLE CATEGORY



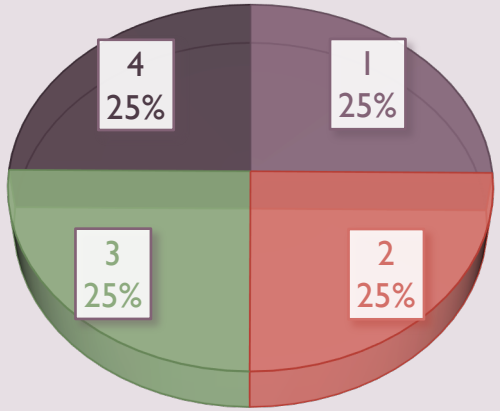
DAYS IN POLICY FROM ACCIDENT



POLICE REPORT FILED

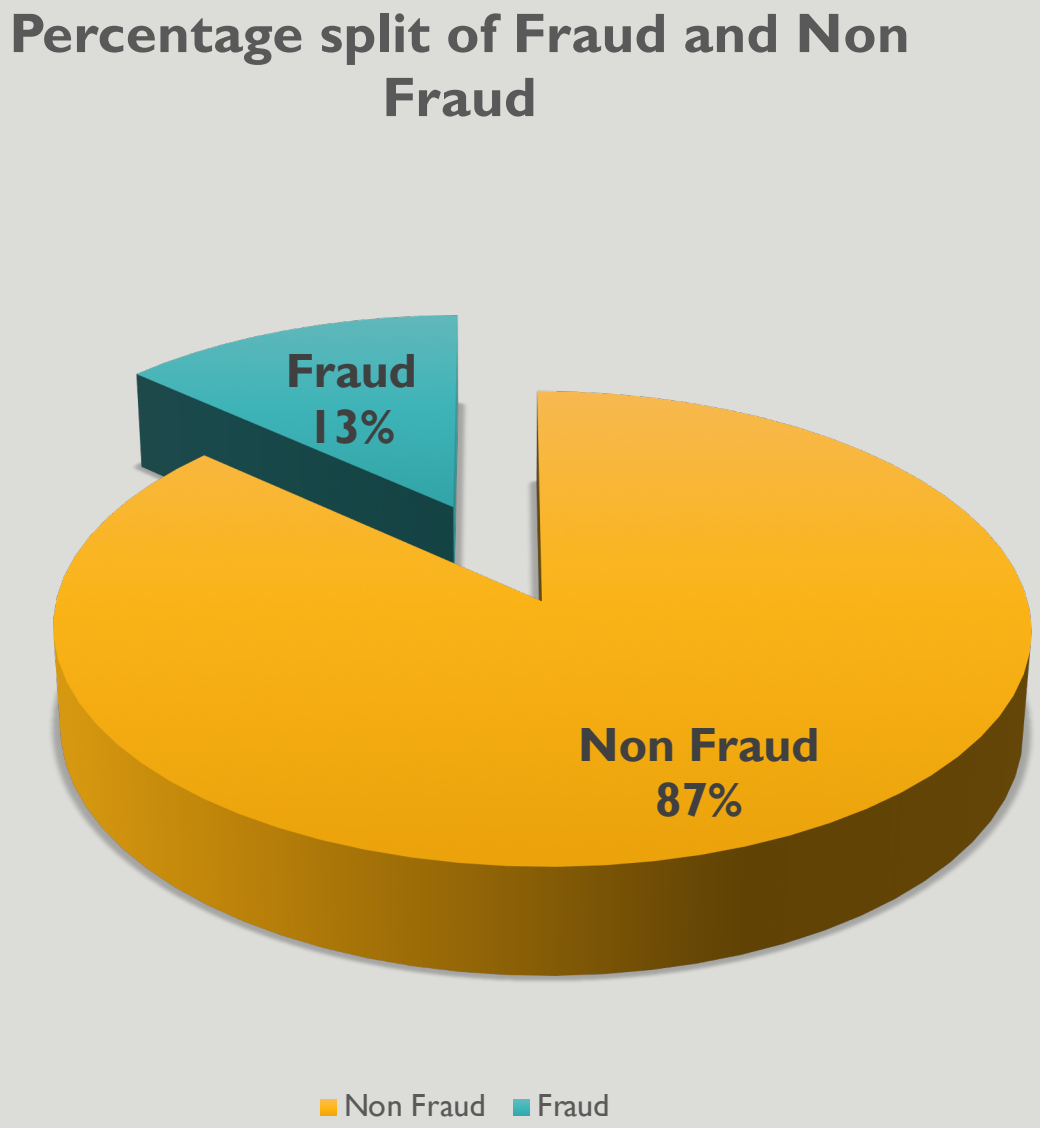


DRIVER RATING

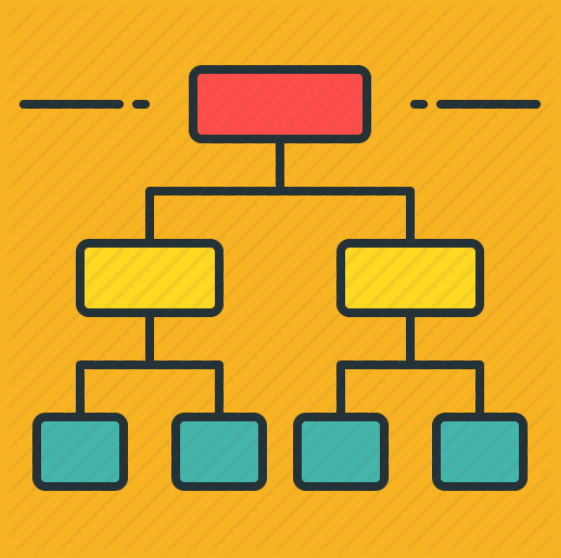


DATA SUMMARY (2)

	AGE	DEDUCTIBLE	FRAUDFOUND
count	2999.000000	2999.000000	2999.000000
mean	40.055352	407.302434	0.133044
std	13.497026	41.847258	0.339679
min	0.000000	300.000000	0.000000
25%	31.000000	400.000000	0.000000
50%	38.000000	400.000000	0.000000
75%	49.000000	400.000000	0.000000
max	80.000000	700.000000	1.000000



UNBALANCED DATASET



DECISION TREE ALGORITHM

DEFAULT DECISION TREE MODEL

```
#Make default decision tree using training dataset
clf = DecisionTreeClassifier()
clf.fit(X_train, Y_train)

#Prediction on training data
pred=pd.DataFrame(clf.predict(X_train),columns=["Prediction"])
print("--------------------------Decision Tree - Training Data--------------------
print("Accuracy Score on training data using Decision Tree:",accuracy_score(Y_train,
print("Confusion Matrix on training data using Decision Tree\n", confusion_matrix(Y_

# prediction on test data
pred=pd.DataFrame(clf.predict(X_test),columns=["Prediction"])
print("--------------------------Decision Tree - Test Data-----------------------------------------------\n")
print("Accuracy Score on test data using Decision Tree:",accuracy_score(Y_test,pred["Prediction"]))
print("Balanced Accuracy Score on test data using Decision Tree:",balanced_accuracy_score(Y_test,pred["Prediction"]))
print("Confusion Matrix on test data using Decision Tree\n", confusion_matrix(Y_test,pred["Prediction"]))
print("Classification report\n", classification_report(Y_test,pred["Prediction"]))
```

Default Parameters:

- Criterion: Gini
- Max depth of tree: None
- Min samples in leaf node: 1
- Min samples in node to split: 2

CONFUSION MATRIX ON TEST DATA

	Predicted Non Fraud	Predicted Fraud
Actual Non Fraud	11097	1323
Actual Fraud	50	448

True Positive

False Positive

False Negative

True Negative

Recall of who will purchase ($\frac{448}{(448+50)}$) = 90%

HYPERPARAMETER TUNING

RANDOM AND GRID SEARCH

```
#Hyperparameter tuning done for decision tree classifier
#RANDOM SEARCH----------------------------------------

print("RandomizedSearchCV-Decision tree")
parameters={ 'min_samples_split' : range(5,30,5), 'criterion':['gini','entropy'], 'max_depth': range(30,120,10)}
clf_random = RandomizedSearchCV(clf,parameters,n_iter=30, cv=8,scoring ="recall")
clf_random.fit(X_train, Y_train)
rand_parm=clf_random.best_params_
print(rand_parm)

#GRID SEARCH-------------------------------------
print("GridSearchCV-Decision tree")
clf_grid = GridSearchCV(clf,parameters,scoring ="recall")
clf_grid.fit(X_train, Y_train)
grid_parm=clf_grid.best_params_
print(grid_parm)
```

Parameters for variation:

- Max depth of tree: (30,120,10)
- Min samples in the node for split: (5,30,5)
- Criterion for splitting: (Gini, Entropy)

Optimize Recall score!

Purpose is to predict most of the actual fraud cases because there is a high cost involved in not predicting them!

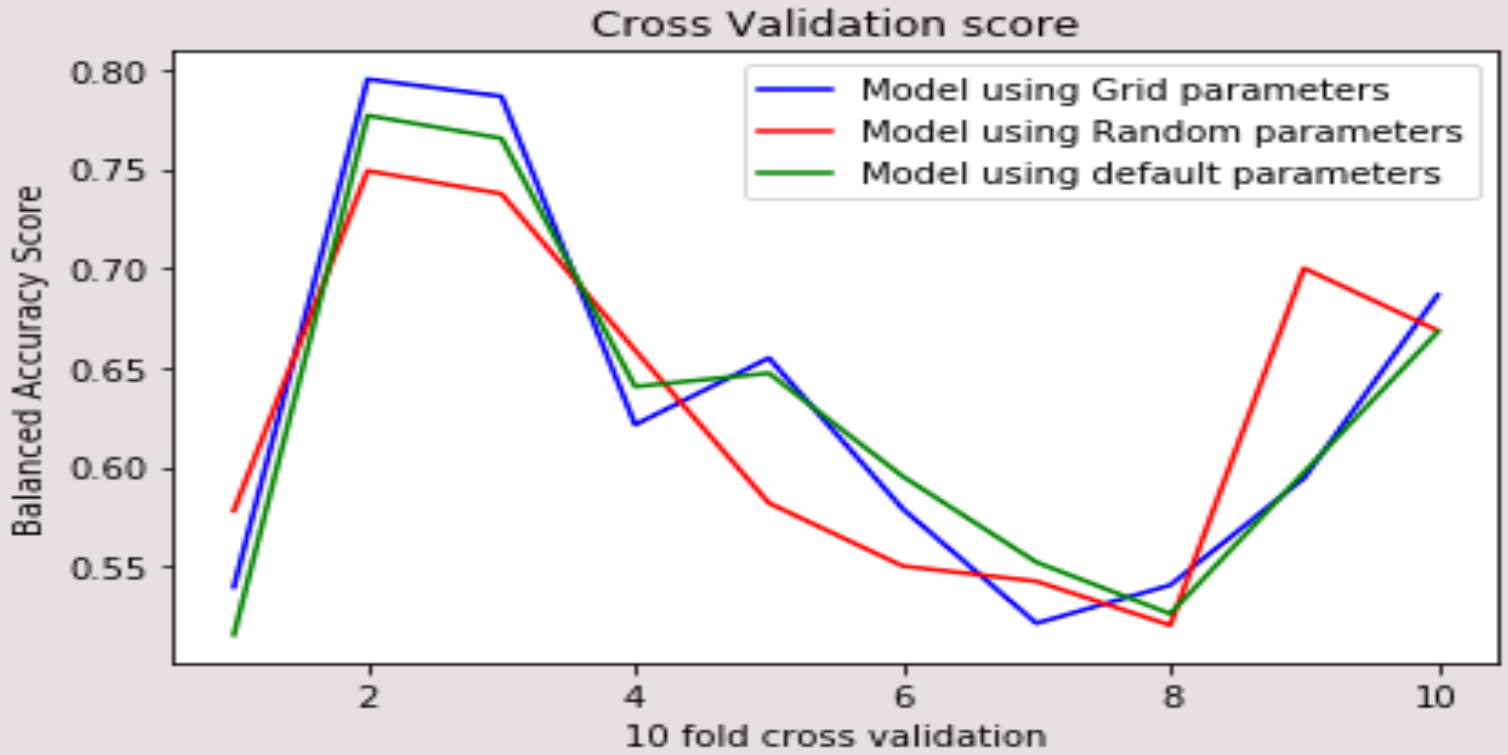
```
RandomizedSearchCV-Decision tree
{'min_samples_split': 5, 'max_depth': 110, 'criterion': 'entropy'}
GridSearchCV-Decision tree
{'criterion': 'gini', 'max_depth': 100, 'min_samples_split': 5}
```

Optimal Parameters

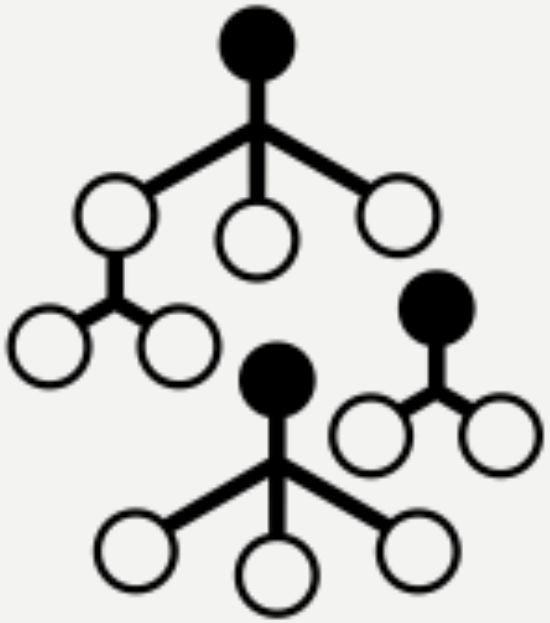
MODEL PERFORMANCE ON TEST DATA

	Default model	Grid Search model	Random Search model
Accuracy	89.39%	89.44%	89.54%
Balanced Accuracy	89.65%	87.28%	86.85%
Precision of fraud cases	25%	25%	25%
Recall of fraud cases	**90%**	85%	84%
F1 – score of fraud cases	39%	38%	38%

CROSS VALIDATION



	Default model	Grid Search model	Random Search model
Mean balanced accuracy score of cross validation	63.18%	62.85%	62.83%



RANDOM FOREST ALGORITHM

DEFAULT RANDOM FOREST MODEL

```
#Random forest
rfc = RandomForestClassifier()
rfc.fit(X_train, Y_train)

#Prediction on training data
pred=pd.DataFrame(rfc.predict(X_train),columns=["Prediction"])
print("--------------------------Random Forest: Training Data----------------------
print("Accuracy Score on training data using Random Forest:",accuracy_score(Y_train,pr
print("Confusion Matrix on training data using Random Forest\n", confusion_matrix(Y_tr

# prediction on test data
pred=pd.DataFrame(rfc.predict(X_test),columns=["Prediction"])
print("--------------------------Random Forest: Test Data-----------------------------------------------\n")
print("Accuracy Score on test data using Random Forest:",accuracy_score(Y_test,pred["Prediction"]))
print("Confusion Matrix on test data using Random Forest\n", confusion_matrix(Y_test,pred["Prediction"]))
print("Classification report on test data using Random Forest\n", classification_report(Y_test,pred["Prediction"]))
```

Default Parameters:

- Max features: Auto
- Number of tress: 10
- Criterion: Gini
- Max depth of tree: None
- Min samples in leaf node: 1
- Min samples in node to split: 2

HYPERPARAMETER TUNING

RANDOM AND GRID SEARCH

```
#Hyperparameter tuning done for random forest classifier
#RANDOM SEARCH----------------------------------------

print("RandomizedSearchCV-Decision tree")
parameters={'max_features': range(10,50,5) ,'n_estimators':[25,30,40,50], 'criterion':['gini','entropy']}
rfc_random = RandomizedSearchCV(rfc,parameters,n_iter=30,cv=5, scoring ='precision')
rfc_random.fit(X_train, Y_train)
rand_parm_rfc=rfc_random.best_params_
print(rand_parm_rfc)

#GRID SEARCH-------------------------------------
print("GridSearchCV-Decision tree")
rfc_grid = GridSearchCV(rfc,parameters,scoring ='precision')
rfc_grid.fit(X_train, Y_train)
grid_parm_rfc=rfc_grid.best_params_
print(grid_parm_rfc)
```

Parameters for variation:

- Max features: (10,50,5)
- Number of trees: (25,30,40,50)
- Criterion for splitting: (Gini, Entropy)

Optimize Precision score!

Want to correctly predict more customers who will actually buy!

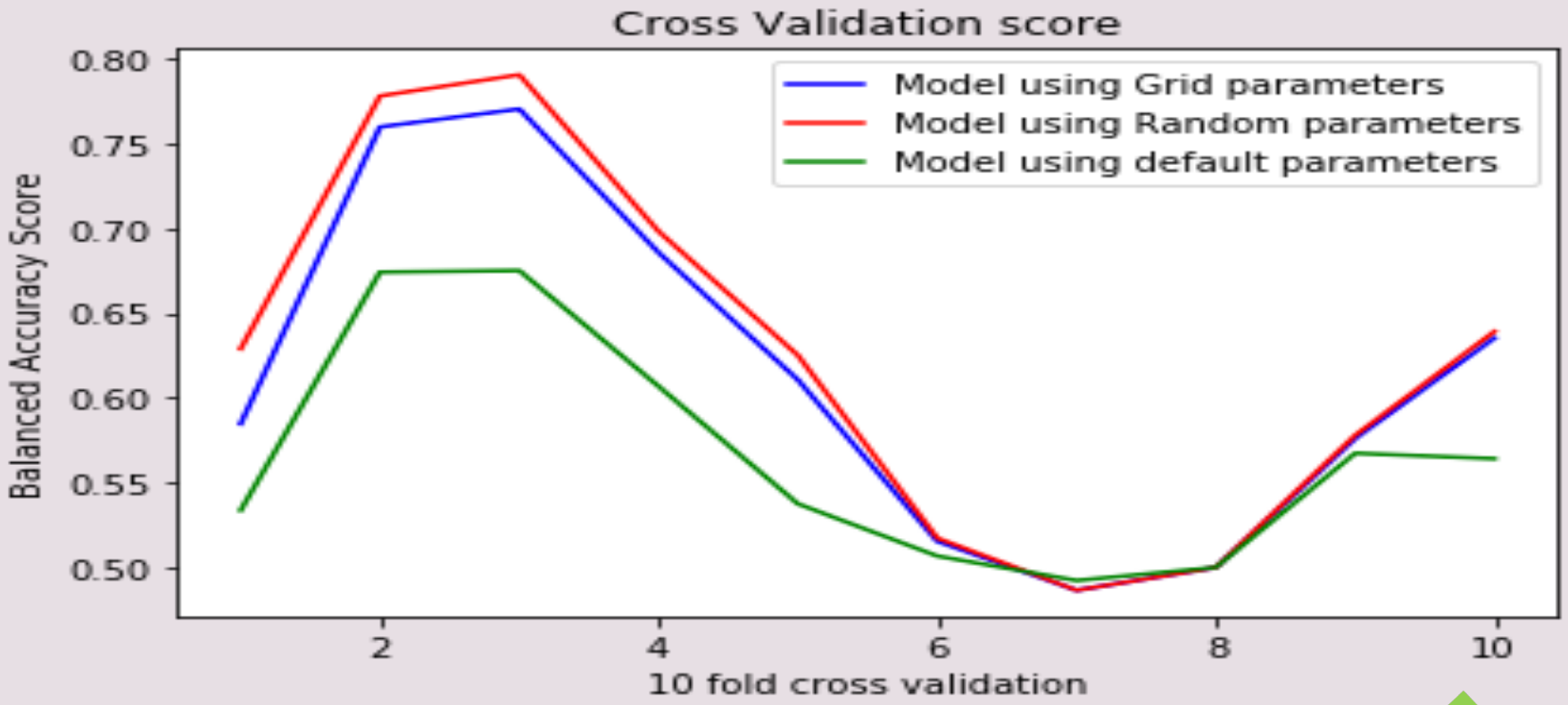
```
RandomizedSearchCV-Decision tree
{'n_estimators': 40, 'max_features': 10, 'criterion': 'entropy'}
GridSearchCV-Decision tree
{'criterion': 'entropy', 'max_features': 10, 'n_estimators': 50}
```

Optimal Parameters

MODEL PERFORMANCE ON TEST DATA

	Default model	Grid Search model	Random Search model
Accuracy	95.37%	92.96%	92.99%
Balanced Accuracy	86.22%	90.17%	89.7%
Precision of fraud cases	44%	34%	34%
Recall of fraud cases	76%	**87%**	86%
F1 – score of fraud cases	56%	49%	49%

CROSS VALIDATION



Mean balanced accuracy score of cross validation

Default model	Grid Search model	Random Search model
56.57%	61.23%	**62.41%**

Overall Decision tree with default parameters is working best on test data with maximum recall – 90%!

LEARNINGS FROM ASSIGNMENT

Hyperparameter tuning technique to get better model performance

Random forest algorithm

Cross validation