Main Objective of Analysis:

Using various classification models, the goal of this analysis is to determine which model provided the best results based on standard metrics to determine customer churn. This classification analysis would be useful to a managers to determine main reasons why certain customers churn and assess what kinds of factors would prevent future customers from churning.

Data description :

- We use a customer churn dataset from the telecom industry which includes customer data, usage of long-distance, data usage, monthly revenue, type of offerings, and other services purchased by customers. The data, based on a fictional telecom firm.

- Data contains 7043 data point , 21 columns including Target '**churn_value**' column
- Data types summary :

  object 13
  int64 6
  float64 2

-

	0	1	2	3
id	8779-QRDMV	7495-OOKFY	1658-BYGOY	4598-XLKNJ
months	1	8	18	25
offer	None	Offer E	Offer D	Offer C
phone	No	Yes	Yes	Yes
multiple	No	Yes	Yes	No
internet_type	DSL	Fiber Optic	Fiber Optic	Fiber Optic
gb_mon	8	17	52	12
security	No	No	No	No
backup	No	Yes	No	Yes
protection	Yes	No	No	Yes
support	No	No	No	No
unlimited	No	Yes	Yes	Yes
contract	Month-to-Month	Month-to-Month	Month-to-Month	Month-to-Month
paperless	Yes	Yes	Yes	Yes
payment	Bank Withdrawal	Credit Card	Bank Withdrawal	Bank Withdrawal
monthly	39.65	80.65	95.45	98.50
total_revenue	59.65	1024.10	1910.88	2995.07
satisfaction	3	3	2	2
churn_value	1	1	1	1
churn_score	91	69	81	88
cltv	5433	5302	3179	5337

```
id                object
months             int64
offer             object
phone             object
multiple          object
internet_type     object
gb_mon             int64
security          object
backup            object
protection        object
support           object
unlimited         object
contract          object
paperless         object
payment           object
monthly          float64
total_revenue    float64
satisfaction       int64
churn_value        int64
churn_score        int64
cltv               int64
dtype: object
```

EDA , Data cleaning and feature engineering:

- Notice that the data contains a unique ID, an indicator for phone customer status, total lifetime value, total revenue, and a bank-estimated churn score. We will not be using these features, so they can be dropped from the data.

  The data now looks like this :

months	int64
offer	object
multiple	object
internet_type	object
gb_mon	int64
security	object
backup	object
protection	object
support	object
unlimited	object
contract	object
paperless	object
payment	object
monthly	float64
satisfaction	int64
churn_value	int64

- Categorizing the features into categorical, numeric, binary, and ordinal variables. As per the unique values counts we can categorize features into:

  1- Binary_variables : 'multiple', 'security', 'backup', 'protection', 'support', 'unlimited', 'paperless', 'churn_value'

  2- categorical_variables : 'offer', 'internet_type', 'payment'

  3- ordinal_variables : 'contract', 'satisfaction', 'months'

  4- numeric_variables : 'monthly', 'gb_mon'

Variable	Unique Values
months	72
offer	6
multiple	2
internet_type	4
gb_mon	50
security	2
backup	2
protection	2
support	2
unlimited	2
contract	3
paperless	2
payment	3
monthly	1585
satisfaction	5
churn_value	2

- Transform features according to their category:

  For categorical_variables ,one hot encoding is used

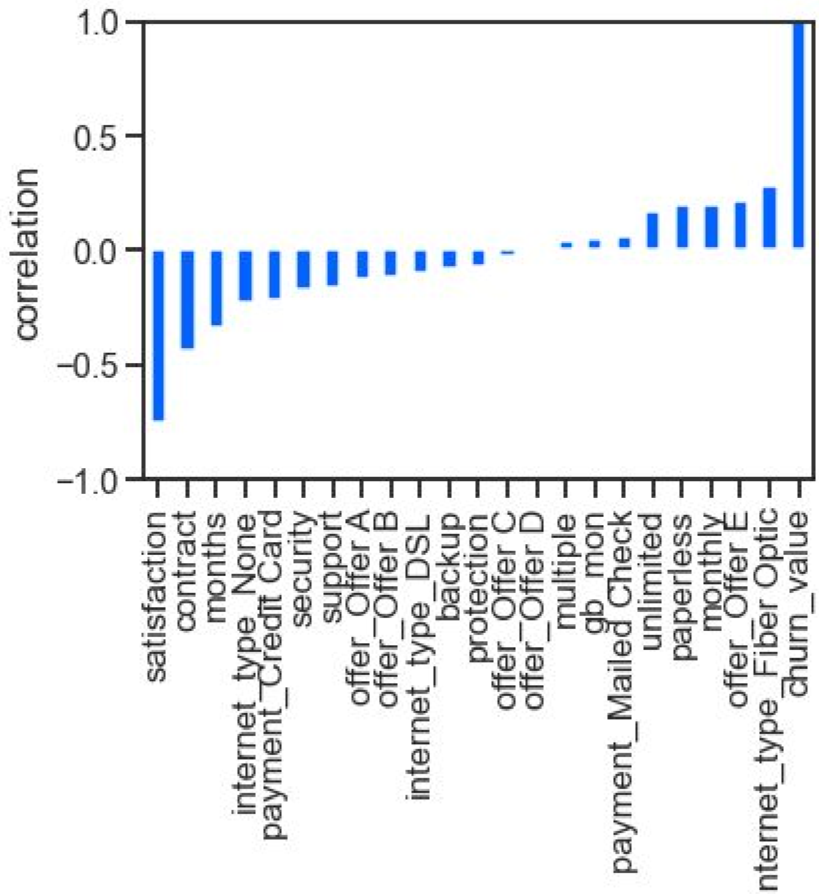
  For Binary_variables LabelBinarizer is used

  For ordinal_variables LabelEncoder is used

- Scaling the data using MinMaxScaler

- Correlation between features and target:

satisfaction	-0.754649
contract	-0.435398
months	-0.337205
internet_type_None	-0.227890
payment_Credit Card	-0.218528
security	-0.171226
support	-0.164674
offer_Offer A	-0.126654
offer_Offer B	-0.117723
internet_type_DSL	-0.099716
backup	-0.082255
protection	-0.066160
offer_Offer C	-0.020660
offer_Offer D	0.001435
multiple	0.040102
gb_mon	0.048868
payment_Mailed Check	0.056348
unlimited	0.166545
paperless	0.191825
monthly	0.193356
offer_Offer E	0.214648
internet_type_Fiber Optic	0.279623
churn_value	1.000000



- From the data we see it's unbalanced data:

0 : 0.73463

1 : 0.26537

So StratifiedShuffleSplit will be used to split the data

Comparing Models:

1- LogisticRegression Model :

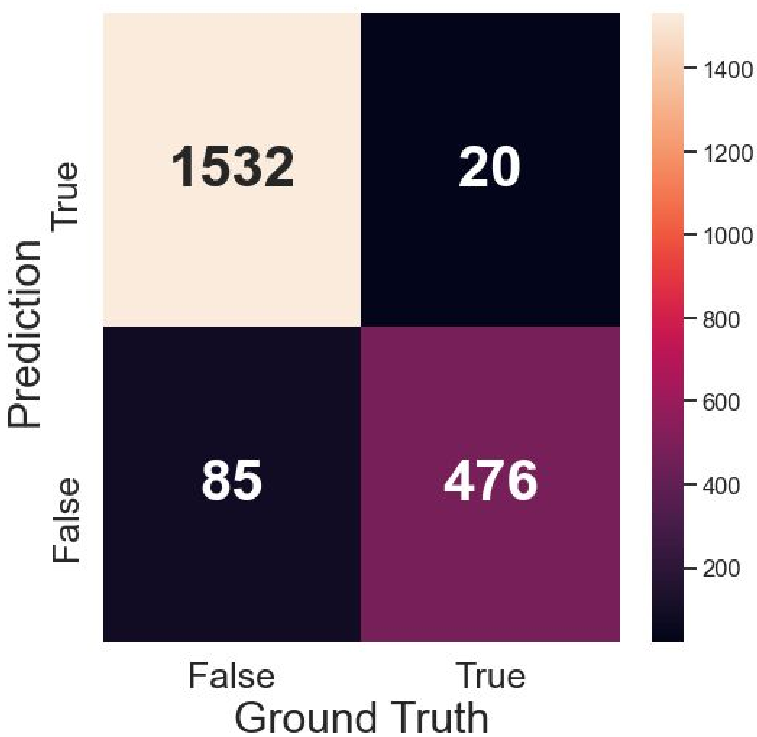
Accuracy score: 0.95
F1 Score: 0.9

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      1552
           1       0.96      0.85      0.90       561

    accuracy                           0.95      2113
   macro avg       0.95      0.92      0.93      2113
weighted avg       0.95      0.95      0.95      2113

Accuracy score:  0.95
F1 Score:  0.9
```

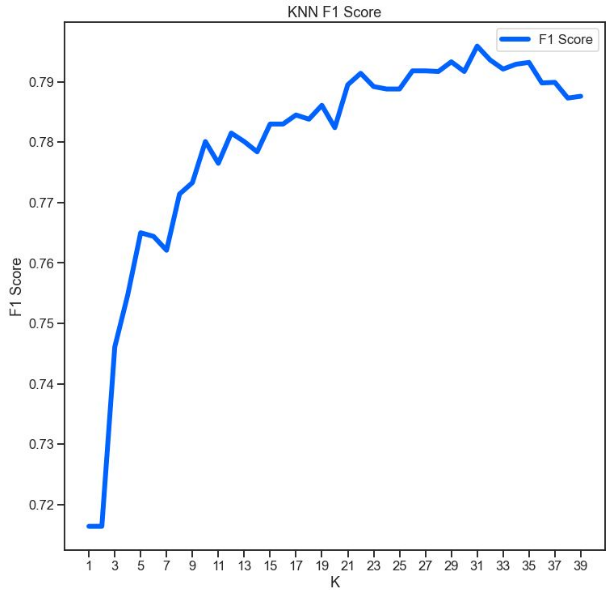


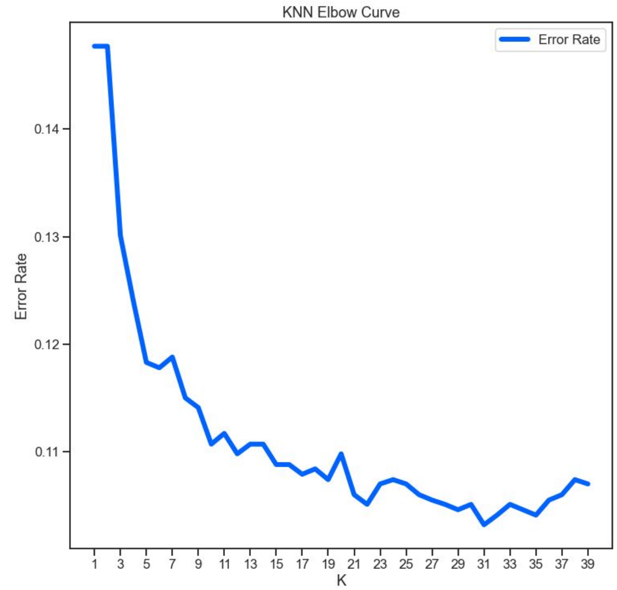
2- KNeighborsClassifier Model:

By examining results for values of K from 1 to 40 to determine the best k these are the results for k=30:

Accuracy score: 0.89

F1 Score: 0.77



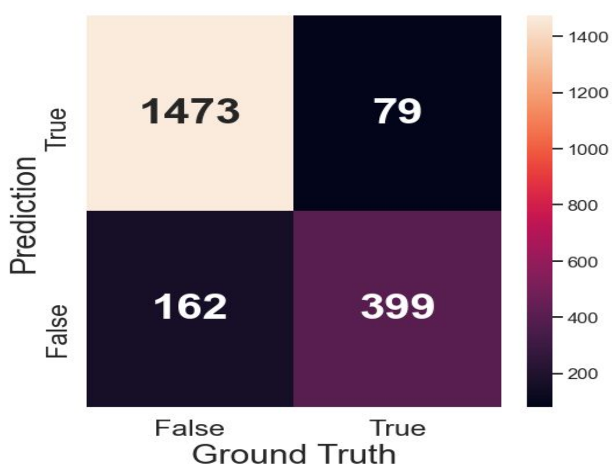



```
              precision    recall  f1-score   support

           0       0.90      0.95      0.92      1552
           1       0.83      0.71      0.77       561

    accuracy                           0.89      2113
   macro avg       0.87      0.83      0.85      2113
weighted avg       0.88      0.89      0.88      2113

Accuracy score:  0.89
F1 Score:  0.77
```



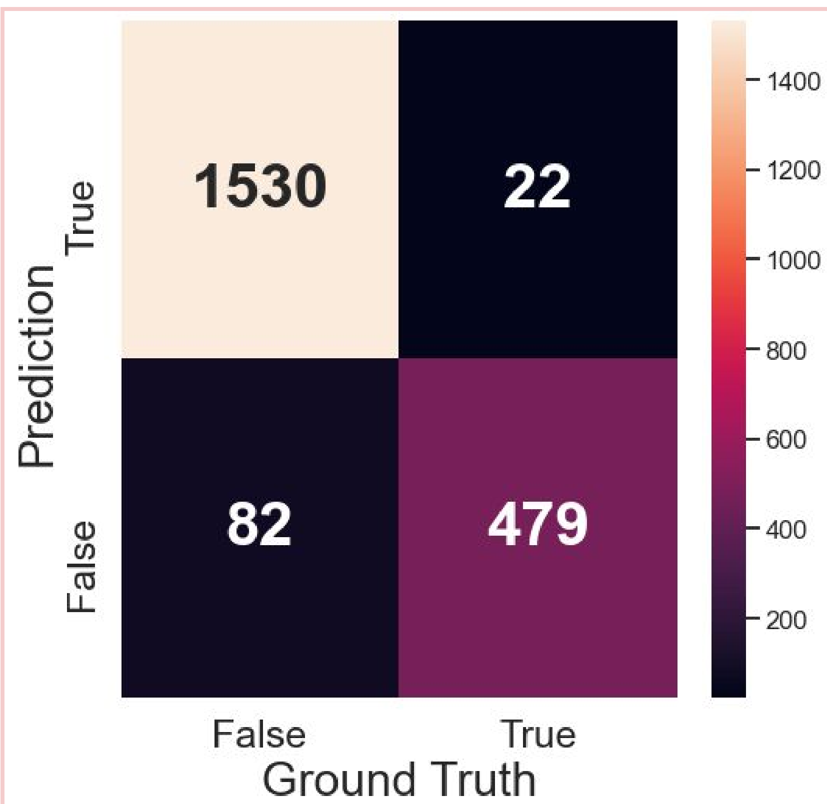
3- LinearSVC :

Accuracy score: 0.95

F1 Score: 0.9

	precision	recall	f1-score	support
0	0.95	0.99	0.97	1552
1	0.96	0.85	0.90	561
accuracy			0.95	2113
macro avg	0.95	0.92	0.93	2113
weighted avg	0.95	0.95	0.95	2113

Accuracy score: 0.95
F1 Score: 0.9



4- DecisionTreeClassifier Model :

For this model max_depth = 20 , node_count =537

Accuracy score: 0.92

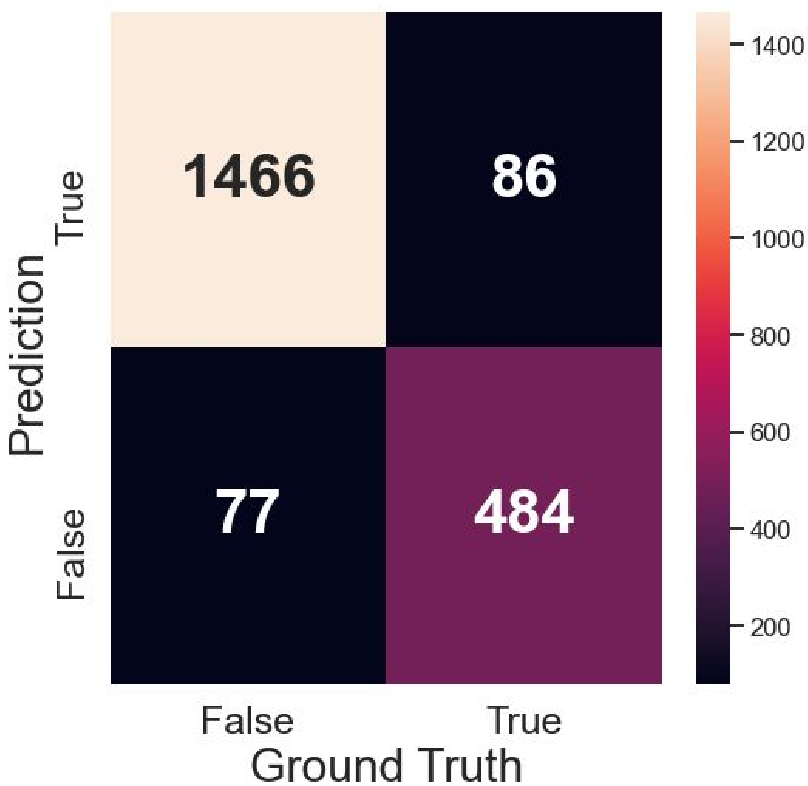
F1 Score: 0.86

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      1552
           1       0.85      0.86      0.86       561

    accuracy                           0.92      2113
   macro avg       0.90      0.90      0.90      2113
weighted avg       0.92      0.92      0.92      2113

Accuracy score:   0.92
F1 Score:   0.86
```



5- DecisionTreeClassifier Model using GridSearchCV :

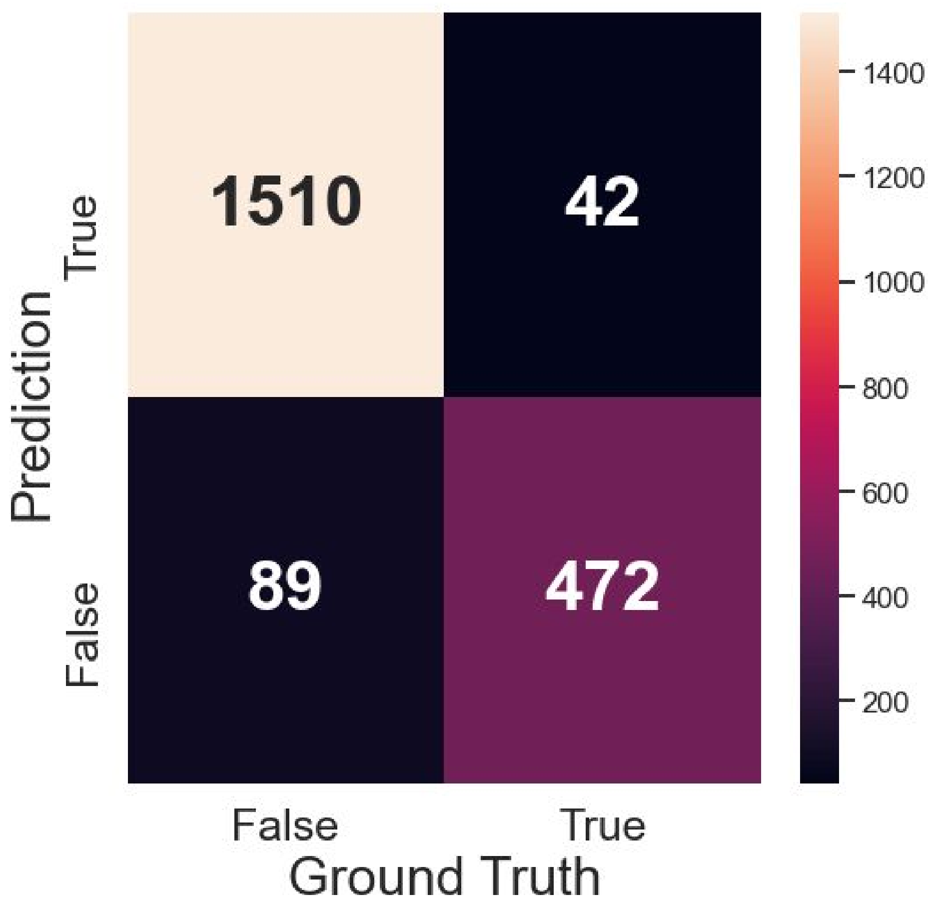
For best estimator : max_depth = 57 , node_count =7

Accuracy score: 0.94

F1 Score: 0.88

	precision	recall	f1-score	support
0	0.94	0.97	0.96	1552
1	0.92	0.84	0.88	561
accuracy			0.94	2113
macro avg	0.93	0.91	0.92	2113
weighted avg	0.94	0.94	0.94	2113

```
Accuracy score:  0.94
F1 Score:  0.88
```



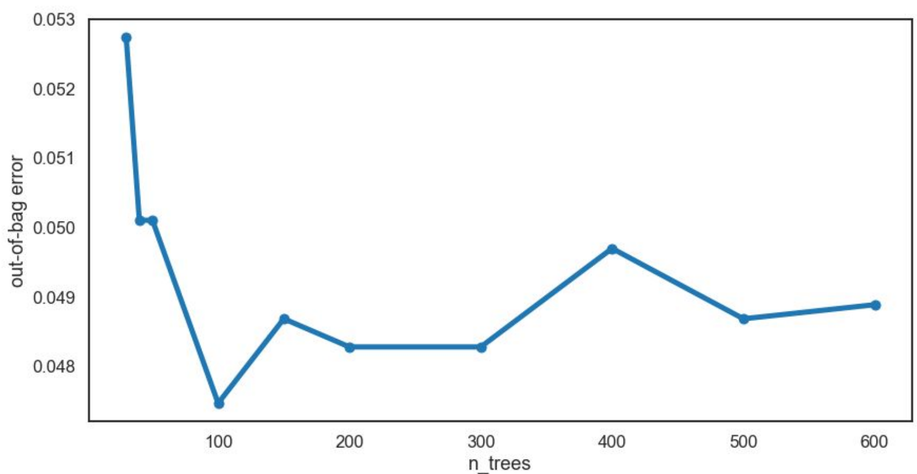
6-RandomForestClassifier Model :

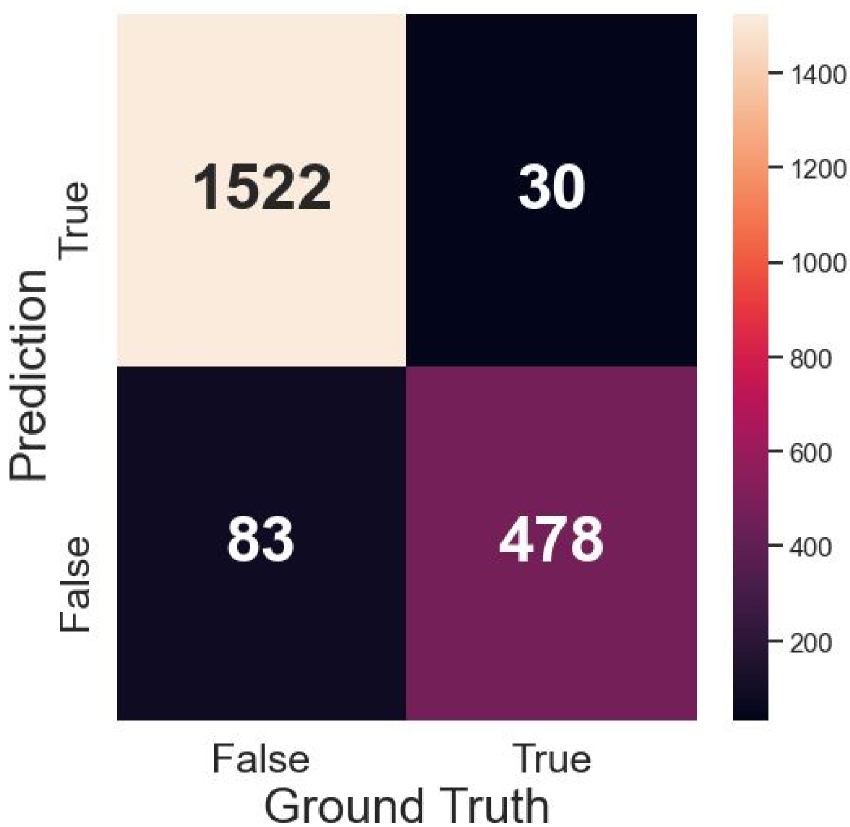
Using RandomForestClassifier as bagging example, and trying different number of trees [30, 40, 50, 100, 150, 200, 300, 400 ,500 ,600] to find one with lowest error :

Accuracy score: 0.95

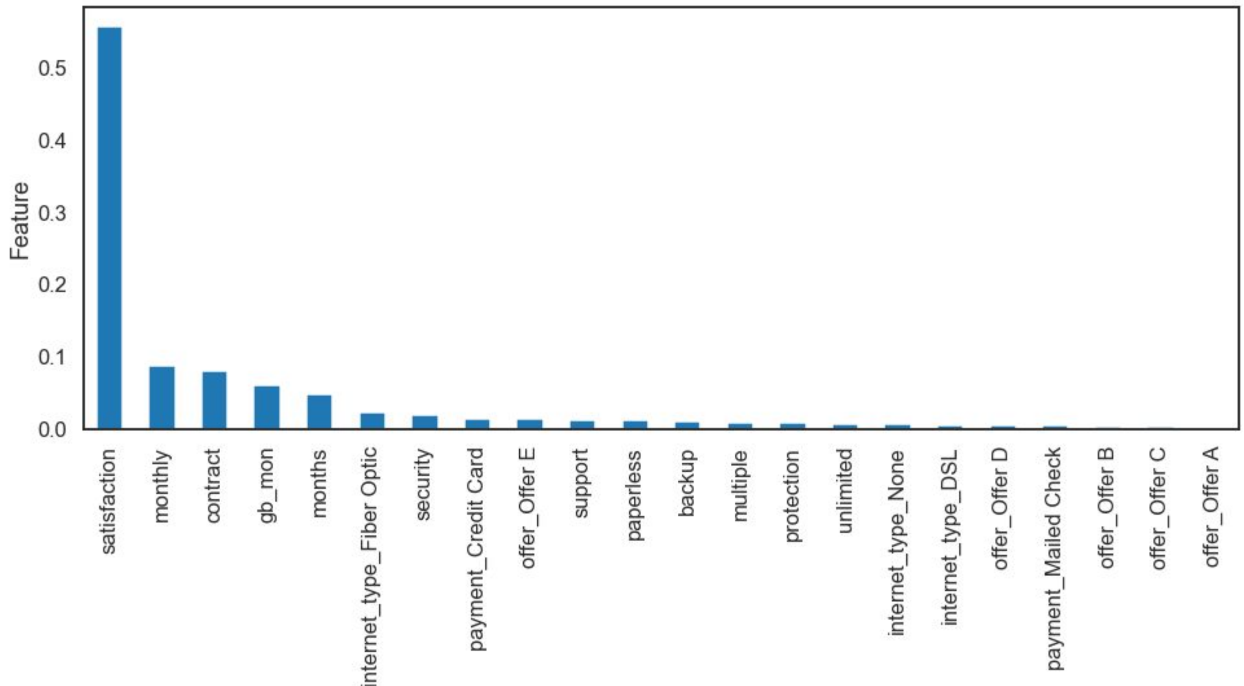
F1 Score: 0.89

	oob
n_trees	
30.0	0.052738
40.0	0.050101
50.0	0.050101
100.0	0.047465
150.0	0.048682
200.0	0.048276
300.0	0.048276
400.0	0.049696
500.0	0.048682
600.0	0.048884





Feature importance figure:



	precision	recall	f1-score	support
0	0.95	0.98	0.96	1552
1	0.94	0.85	0.89	561
accuracy			0.95	2113
macro avg	0.94	0.92	0.93	2113
weighted avg	0.95	0.95	0.95	2113

```
Accuracy score:  0.95
F1 Score:  0.89
```

7- VotingClassifier Model:

Using the Logistic Regression Model and the Random forest Model :

Accuracy score: 0.95

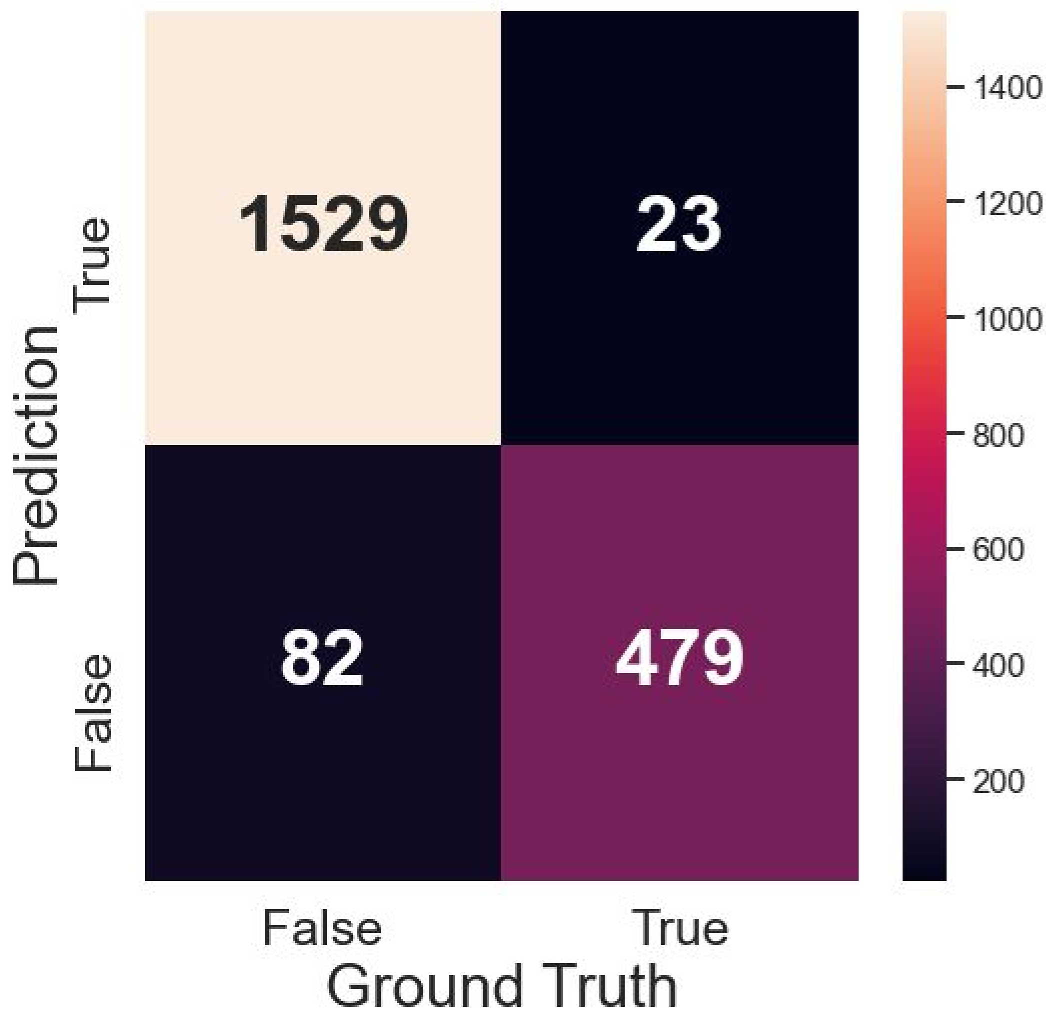
F1 Score: 0.89

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      1552
           1       0.95      0.85      0.90       561

    accuracy                           0.95      2113
   macro avg       0.95      0.92      0.93      2113
weighted avg       0.95      0.95      0.95      2113

Accuracy score:  0.95
F1 Score:  0.89
```



Recommendations:

For the data in hand , LogisticRegression and LinearSVC Models seem to give the best results regarding accuracy and F1-score.

Comparing time consumed training and predicting using the same data splits

LogisticRegression shows less time taken so it will be more recommended.

```
%%timeit
# Standard logistic regression
lr = LogisticRegression(solver='liblinear').fit(X_train, y_train)
y_lr_predict = lr.predict(X_test)
```

```
32.9 ms ± 269 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

```
%%timeit
LSVC = LinearSVC()
LSVC.fit(X_train, y_train)
y_LSVC_pred = LSVC.predict(X_test)
```

```
55.5 ms ± 1.59 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Key Findings and Insights :

From the data and figures above, satisfaction and contract are greatly affecting the churn value.

Suggestions for next steps :

more research to be done on how to improve the customer satisfaction and gather more information about the factors contributing to it.