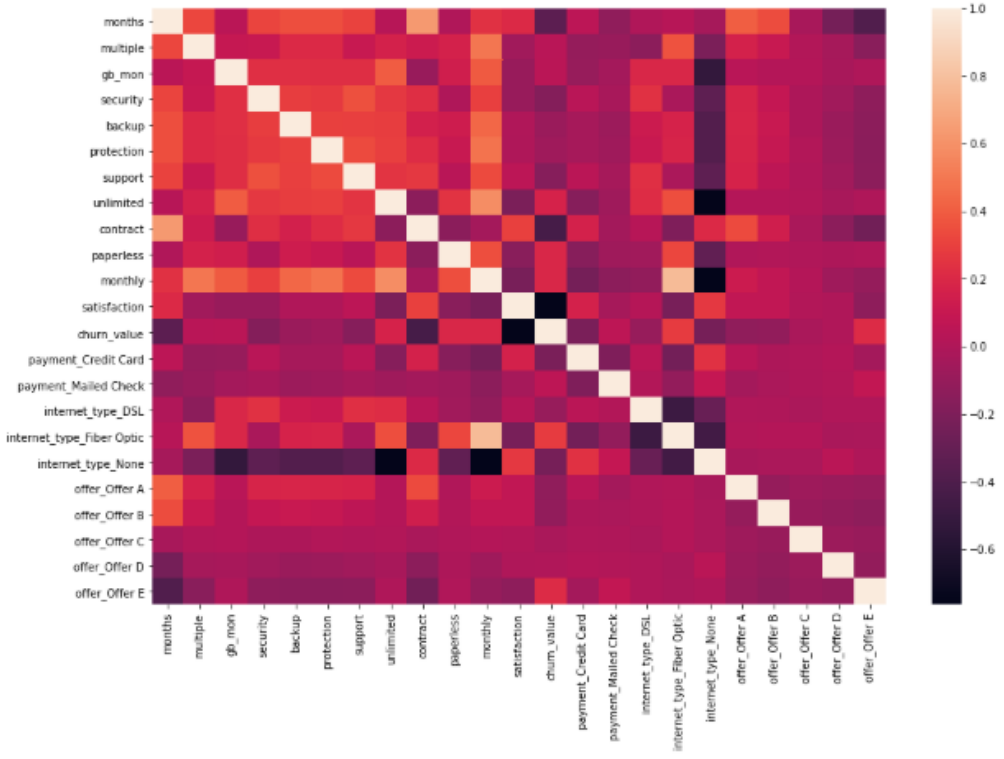
IBM Introduction to Machine Learning

Supervised Learning: Classification

Main objectives and brief description

For this final assignment, models will be focused both on prediction and interpretation. The main objectives are to find patterns, characteristics or correlations that lead customers to stay or abandon a company by deploying different classification models and analyzing which is the best in terms of prediction or interpretation. The python notebook code, the models and further details can be found on: https://github.com/estebancarboni/IBM-Introduction-to-Machine-Learning/blob/master/Supervised%20Learning%20Classification%20Final%20Project.ipynb

Classification models will be deployed from a customer churn dataset from a fictional telecom firm which includes customer data, usage of long-distance, data usage, monthly revenue, type of offerings, and other services purchased by customers. Here we can see a correlation heatmap after data cleaning and feature engineering:



Data exploration

Several Excel files have been combined from the course materials. We are using the subset of customers who have phone accounts. The data include a mix of numeric, categorical, and ordinal variables. These are the features, where “churn_value” is out target variable (whether the customer churned or not). 1869 people, approximately 27% has churned as we can see here.

```
0    5174
1    1869
Name: churn_value, dtype: int64
0    0.73463
1    0.26537
Name: churn_value, dtype: float64
```

```
months                        float64
multiple                        int64
gb_mon                        float64
security                        int64
backup                          int64
protection                      int64
support                         int64
unlimited                       int64
contract                      float64
paperless                       int64
monthly                       float64
satisfaction                  float64
churn_value                     int64
payment_Credit Card             int64
payment_Mailed Check            int64
internet_type_DSL               int64
internet_type_Fiber Optic       int64
internet_type_None              int64
offer_Offer A                   int64
offer_Offer B                   int64
offer_Offer C                   int64
offer_Offer D                   int64
offer_Offer E                   int64
dtype: object
```

Classifier models

We train three different classifier models (in addition to the one trained previously throughout the course): Logistic Regression, K-Nearest Neighbors and Decision Trees. First, we split the data using StratifiedShuffleSplit (test size of 0.3), preserving the percentage of samples for each class, and then check if y_train and y_test are equivalent. This split is used for the three models.

```
y_train:
0    0.734686
1    0.265314
Name: churn_value, dtype: float64
y_test:
0    0.734501
1    0.265499
Name: churn_value, dtype: float64
```

For **Logistic Regression**, we fit the model without any regularization using all the features. Then, using cross validation to determine the hyperparameters, we fit using L1 and L2 regularization. After that, we predict the class and examine the probability for the first 10 values. Metrics were also calculated for each model.

y_pred

	lr	l1	l2
0	1	1	1
1	0	0	0
2	1	1	1
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	1	1	1
8	1	1	1
9	0	0	0

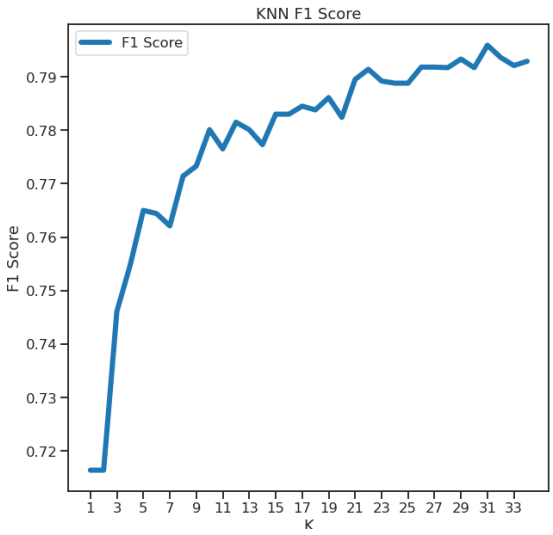
y_prob

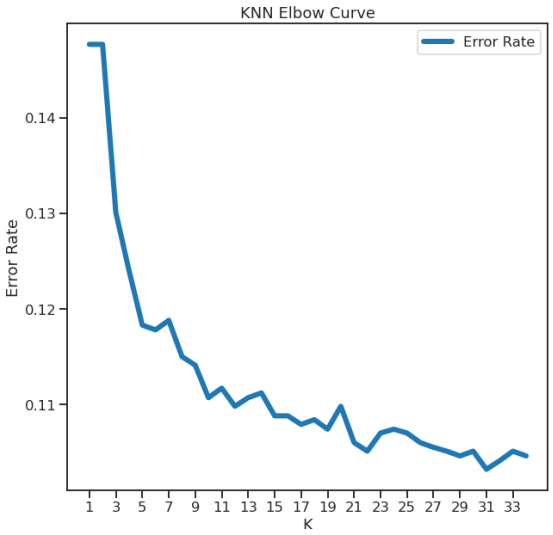
	lr	l1	l2
0	0.998689	1.000000	0.999997
1	0.998944	1.000000	0.999998
2	0.574648	0.614513	0.613905
3	0.986572	0.999753	0.999258
4	0.999134	0.999990	0.999969
5	0.995452	0.998866	0.998685
6	0.998091	0.999976	0.999927
7	0.984110	0.999791	0.999340
8	0.993882	0.999998	0.999980
9	0.998773	0.999986	0.999957

	lr	l1	l2
precision	0.950684	0.952993	0.954013
recall	0.950308	0.953147	0.954094
fscore	0.949289	0.952503	0.953433
accuracy	0.950308	0.953147	0.954094

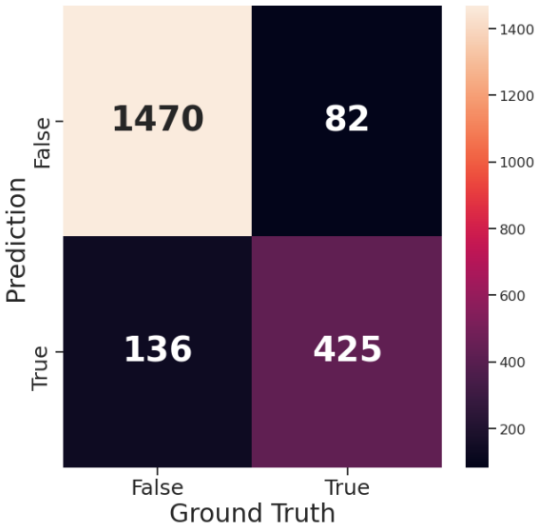
Finally, we plot the confusion matrices:

For deploying a **K-Nearest Neighbors** classifier we need to evaluate which is the right value for K, focusing on two measures: the higher F-1 Score and the lowest Error Rate. After plotting, both results suggest that 31 is the optimal value.





Again, examine the Precision, Recall, F-1 Score, and Accuracy of the classification, and visualize the Confusion Matrix for K = 31.

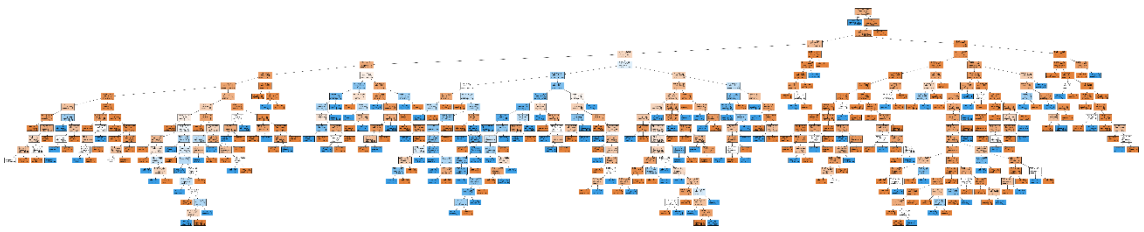


	precision	recall	f1-score	support
0	0.92	0.95	0.93	1552
1	0.84	0.76	0.80	561
accuracy			0.90	2113
macro avg	0.88	0.85	0.86	2113
weighted avg	0.89	0.90	0.90	2113

Accuracy score: 0.9
F1 Score: 0.8

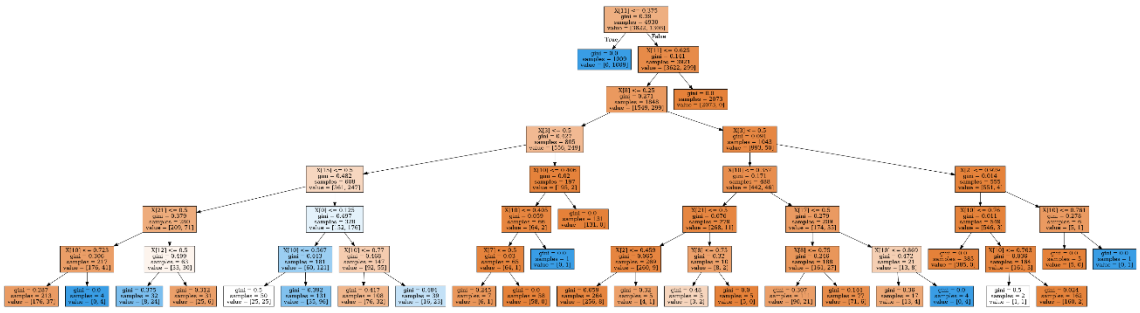
The third model, a **Decision Tree Classifier**, is fitted without limits on maximum depth, features or leaves. We discover 537 nodes and a max depth of 20. Then we search for the metrics both for the training data and test data. The decision tree predicts a little better on the training data than the test data, which is consistent with overfitting.

	train	test
accuracy	0.999391	0.921439
precision	1.000000	0.845884
recall	0.997706	0.860963
f1	0.998852	0.853357



Using grid search with cross validation, we seek for a decision tree that performs well on the test data set. We find 53 nodes and a depth of 7, a simpler tree. These test errors are a little better than the previous ones. It would seem the previous example overfit the data, but only slightly so.

	train	test
accuracy	0.958418	0.938476
precision	0.951679	0.920078
recall	0.888379	0.841355
f1	0.918940	0.878957

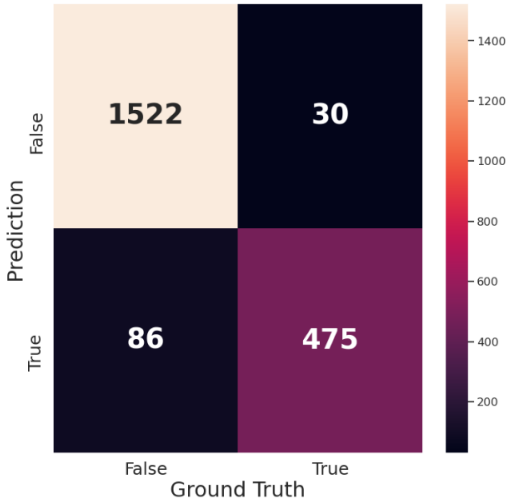


From the fourth model, **Random Forest** classifier, we analyze the results we saw throughout the course.

n_trees	RandomForest	ExtraTrees
15.0	0.056795	0.066126
20.0	0.055984	0.065517
30.0	0.053144	0.060041
40.0	0.050913	0.056592
50.0	0.049087	0.054970
100.0	0.047667	0.050710
150.0	0.049290	0.050913
200.0	0.048073	0.049899
300.0	0.048479	0.050507
400.0	0.048479	0.049696

	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	0.91	0.93	2113
weighted avg	0.94	0.95	0.94	2113

	accuracy	precision	recall	f1	auc
0	0.945102	0.940594	0.846702	0.891182	0.913686



Key findings and recommendation

In terms of accuracy, **Logistic Regression** running with a L2 penalty seems the best classificatory, with better values in all metrics (precision, recall, accuracy, f-1 score) than the rest. In terms of explainability, **Decision Tree** classifier makes it easier to interpret and require no data preprocessing, despite it takes more time to run and tend to overfit and be very sensitive to different data. Cross validation and pruning could help to make it simpler.

Suggestions for next steps

Extending to other models that have noy been analyzed, like Support Vector Machine classifier, as well as splitting data with another method or test size, or dropping features could lead to a slightly better prediction or explanation. In my opinion, the deployed models did a good job. Anyone can suggest or revisit the models to achieve a better explanation or a better prediction: https://github.com/estebancarboni/IBM-Introduction-to-Machine-Learning/blob/master/Supervised%20Learning%20Classification%20Final%20Project.ipynb