Machine Learning 1
Classification Project
By
Arijit Chakrabarti

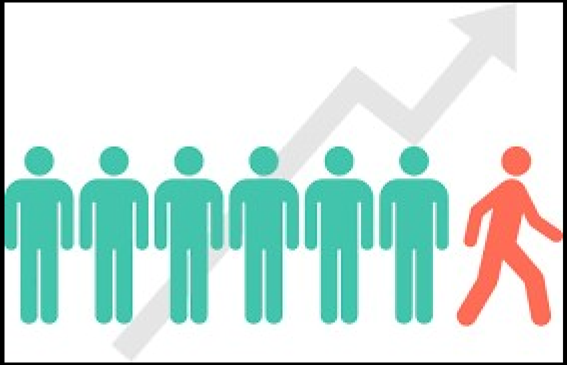
March 2020 Cohort

Selected data-set and objective

- A previous Kaggle competition data-set
- Pertains to Insurance Industry
- Churn prediction model
- Objective to minimise F1 score
- Classification Problem on a total of 15 anonymized features

The Project Flow

- EDA
  - Data was already processed when anonymised
- Identifying the right machine learning model
- Applying strategies to improve scores
- Optimising processing time
- Cross-validation & hyper-parameter tuning
- Fun with Auto ML (Pycaret & Tpot)
- Conclusion and closing remarks



Initial EDA

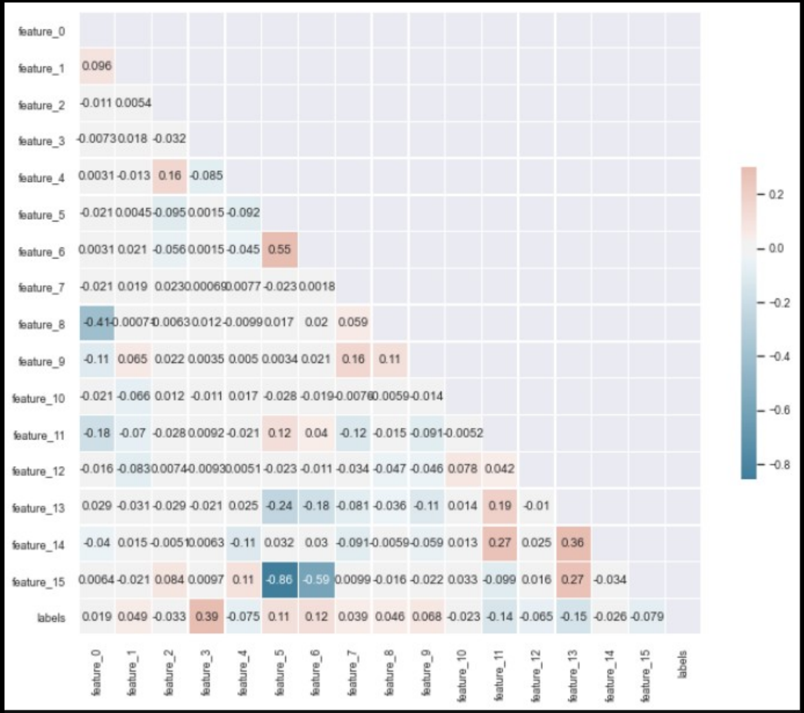
- Shape: (33908,17)
- First 7 features continuous
- Last 9 features categorical
- TV - ‘Labels’

	count	mean	std	min	25%	50%	75%	max
feature_0	33908.0	-0.004158	0.999776	-2.159994	-0.747384	-0.182341	0.665225	5.091402
feature_1	33908.0	0.002584	1.014268	-3.081149	-0.422787	-0.297324	0.022901	33.094776
feature_2	33908.0	-0.000213	1.000872	-1.779108	-0.938003	0.023260	0.624050	1.825628
feature_3	33908.0	-0.000053	1.002512	-1.002478	-0.602517	-0.303517	0.236237	18.094700
feature_4	33908.0	-0.000298	1.003724	-0.569351	-0.569351	-0.246560	0.076230	19.443647
feature_5	33908.0	-0.004652	0.993984	-0.411453	-0.411453	-0.411453	-0.411453	8.127648
feature_6	33908.0	-0.007498	0.802696	-0.251940	-0.251940	-0.251940	-0.251940	23.625644
feature_7	33908.0	4.336381	3.273376	0.000000	1.000000	4.000000	7.000000	11.000000
feature_8	33908.0	1.171051	0.606730	0.000000	1.000000	1.000000	2.000000	2.000000
feature_9	33908.0	1.225345	0.749104	0.000000	1.000000	1.000000	2.000000	3.000000
feature_10	33908.0	0.018137	0.133450	0.000000	0.000000	0.000000	0.000000	1.000000
feature_11	33908.0	0.555503	0.496917	0.000000	0.000000	1.000000	1.000000	1.000000
feature_12	33908.0	0.159667	0.366303	0.000000	0.000000	0.000000	0.000000	1.000000
feature_13	33908.0	0.639407	0.897627	0.000000	0.000000	0.000000	2.000000	2.000000
feature_14	33908.0	5.520497	3.003241	0.000000	3.000000	6.000000	8.000000	11.000000
feature_15	33908.0	2.562375	0.987148	0.000000	3.000000	3.000000	3.000000	3.000000
labels	33908.0	0.116993	0.321417	0.000000	0.000000	0.000000	0.000000	1.000000

Understanding relationships between features

- The target variable labels has mild correlation with feature_3 & feature_7
- Feature_15 has high negative correlation with feature_5 - 0.85
- Feature_15 and feature_6 also have a negative correlation (though not as high) - -0.59
- Feature_8 has correlation with feature_0 - -0.40
- Feature_13 & feature_14 have positive correlation – 0.36

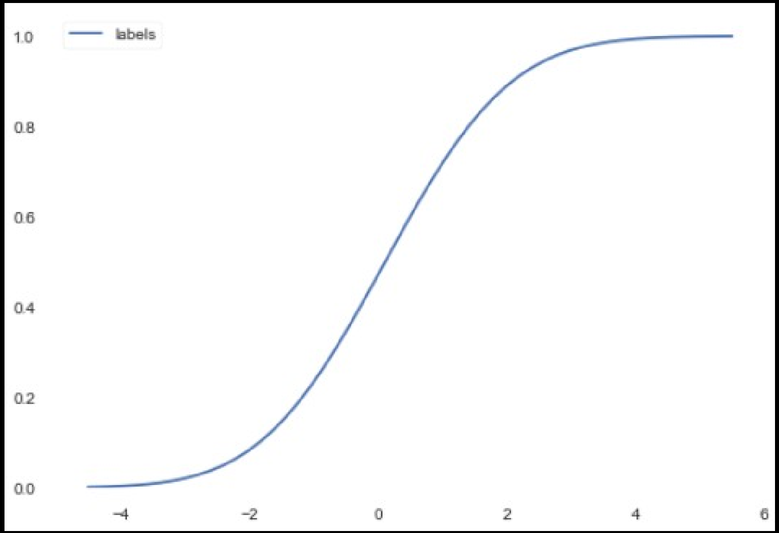
- Interesting to note is that most of the higher correlations are negative in nature except for 1 case - i.e. correlation between feature_13 and feature_14.

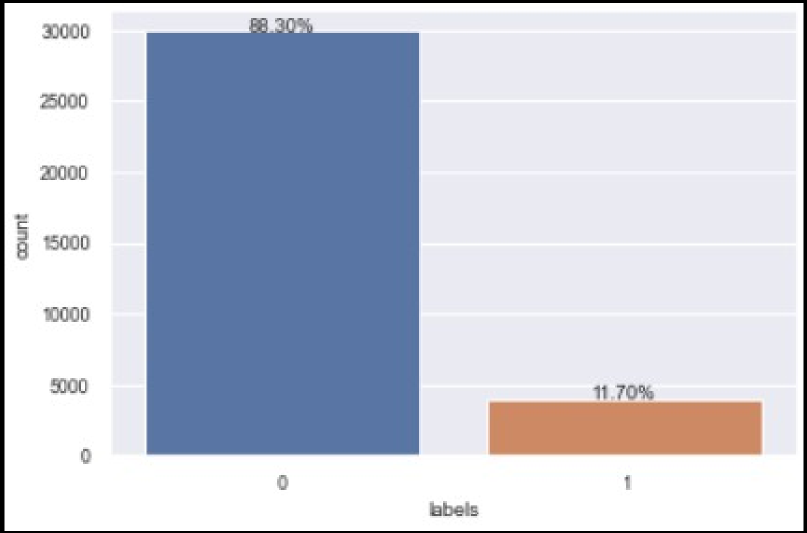


Additional analysis:

- It seems that the data-set has no missing values.
- From the labels column it is clear that the data-set is unbalanced with focus towards 0s rather than 1s.
- The highest range is from column feature_1.
- Columns representing feature-10 through 12 seem to be binary in nature either a yes or a no.

Unbalanced target variable 83% - 0s



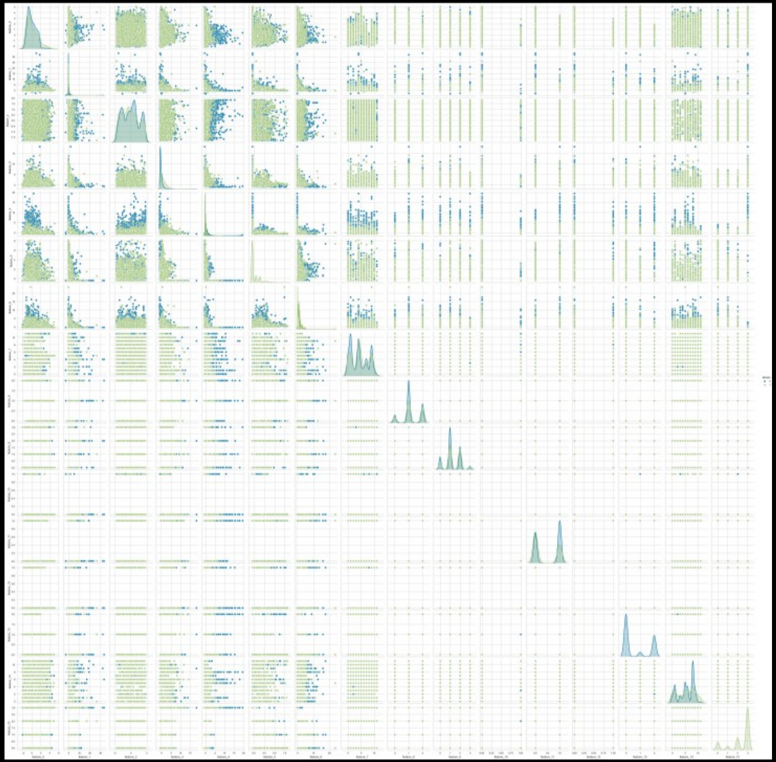


‘Pandas Profiling’ Story

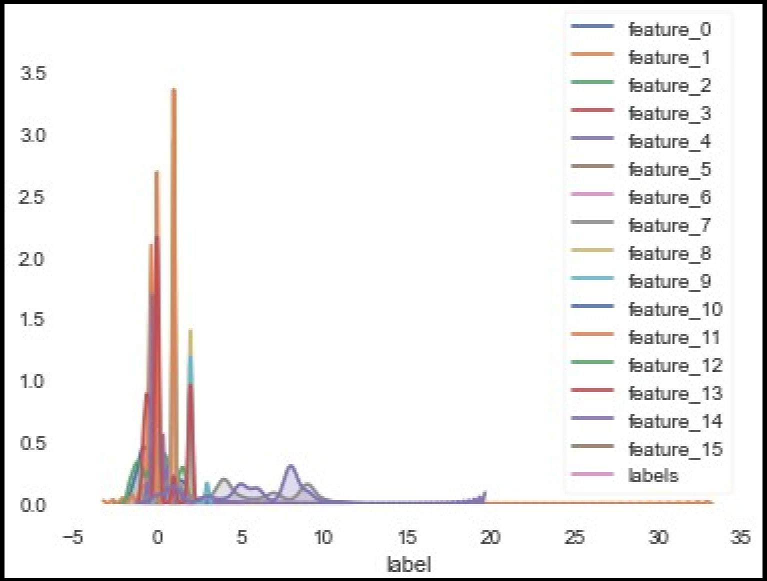
- The data looks clean with no missing cells
- There are two warnings for feature 7 and feature 14 as both have a large share of zeros
  - Firstly they are part of integer columns, and can be ignored on account of encoding categorical data
- The data seems to be normalised on all feature columns
- Possibility of keeping either feature 5 and feature 15 as they have high positive correlation - can be explored while evaluating machine learning strategies

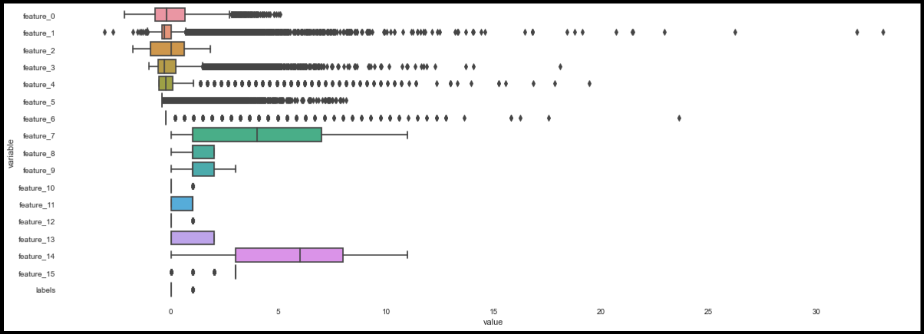
Pair-plot with hue from 'Labels' - TV

- Customers who churn, represented in blue i.e. 1 are clearly clustered towards ends of the data representation.
- Other than the first 6 features the rest of the columns are all encoded from ordinal data.



The data has outliers





- The data clustered around -5 to +5
- Transformations may be needed on feature_14 which has outliers
- Also feature_0 through feature_6 have negative values – may need scaling and other transformations – as it also has high outliers

Models built

- Defined a function to automatically split fit predict and provide model evaluation through classification report on weighted F1 score

- Logistic Regression
- Naive Bayes - though it is known to be a bad estimator - but lets give it a try
- Stochastic Gradient Descent
- K Nearest Neighbours
- Decision Tree
- Random Forest Classifier

Random Forest Classifier had best F1 score

Classification Algorithm	Weighted F1 Train	Weighted F1 Test
Logistic Regression	0.87	0.87
Gaussian NB	0.84	0.84
SGD Classifier	0.87	0.87
KNearest Neighbors	0.91	0.88
Decision Tree	1.00	0.88
Random Forest	1.00	0.89

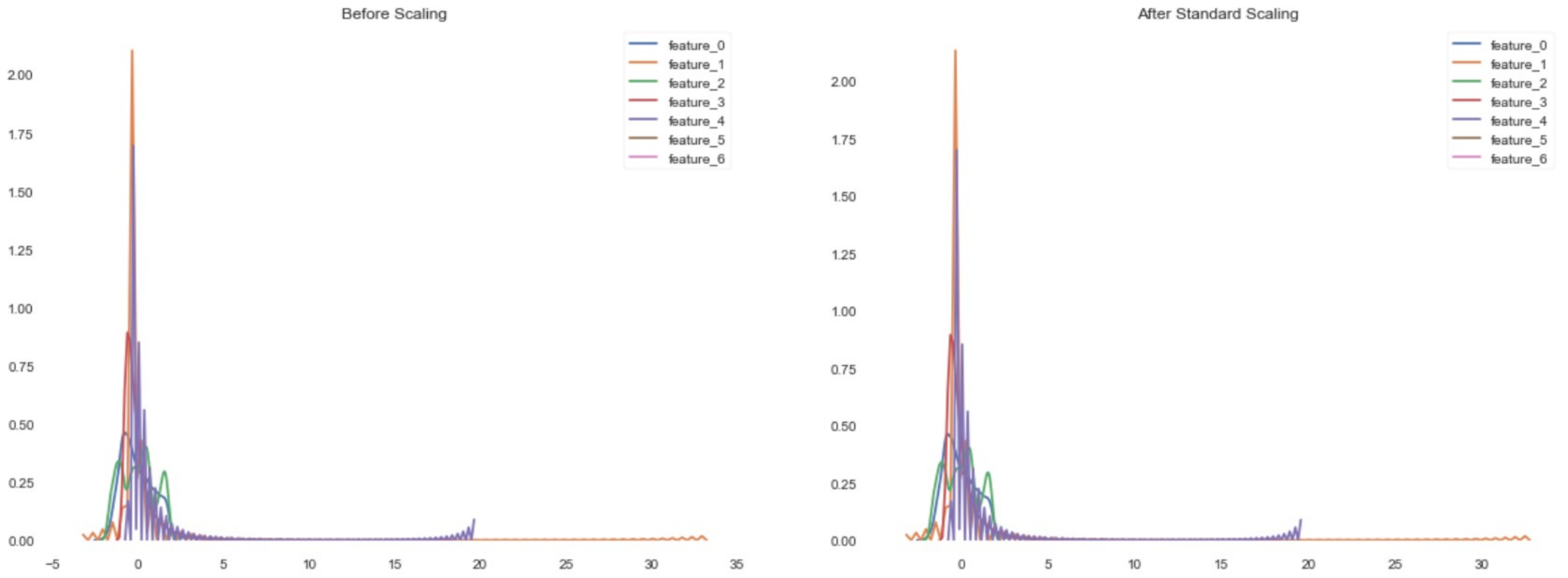
All further techniques applied with this classifier

Score improvement strategies applied

- Standard Scaler
- Robust Scaler
- Normaliser

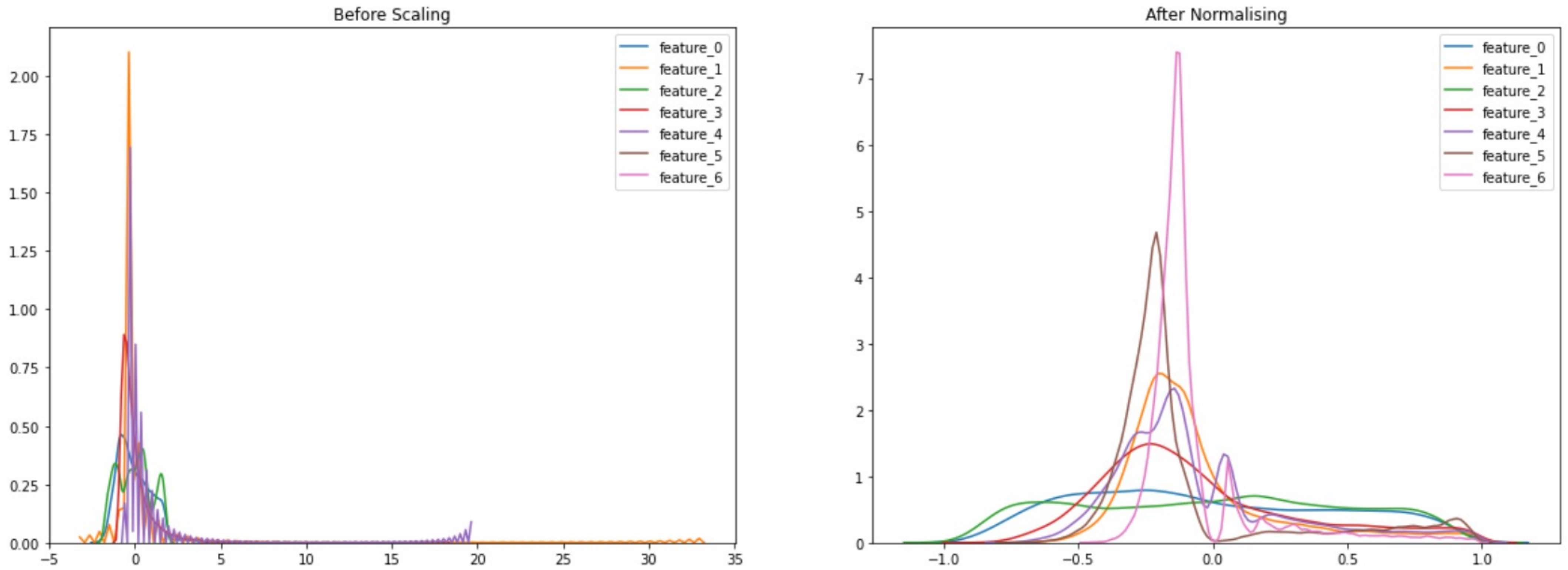
Applied only to the continuous features_0 through feature_6

The data was already standard scaled



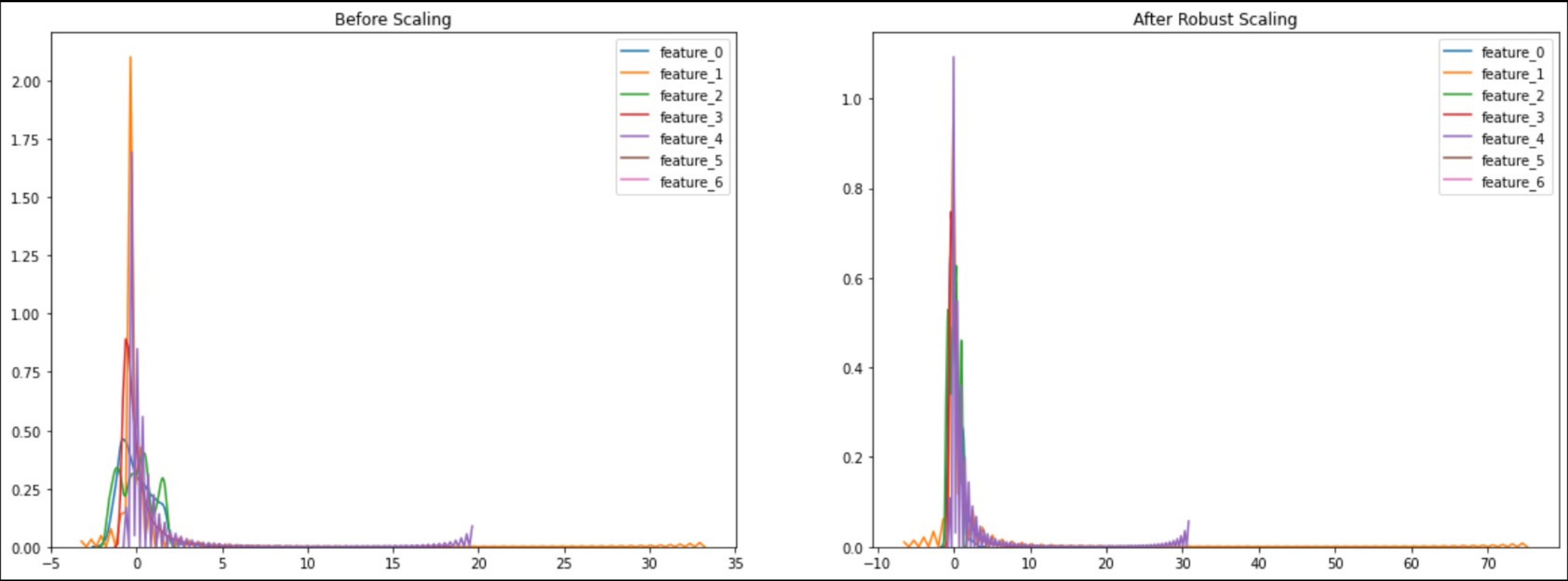
Not tested on machine learning model as there was not change in input

Normalising the data also had minimal effect



Though the input changed dramatically – similar, minuscule 0.01 improvement in F1 score

Robust Scaler handles outliers better



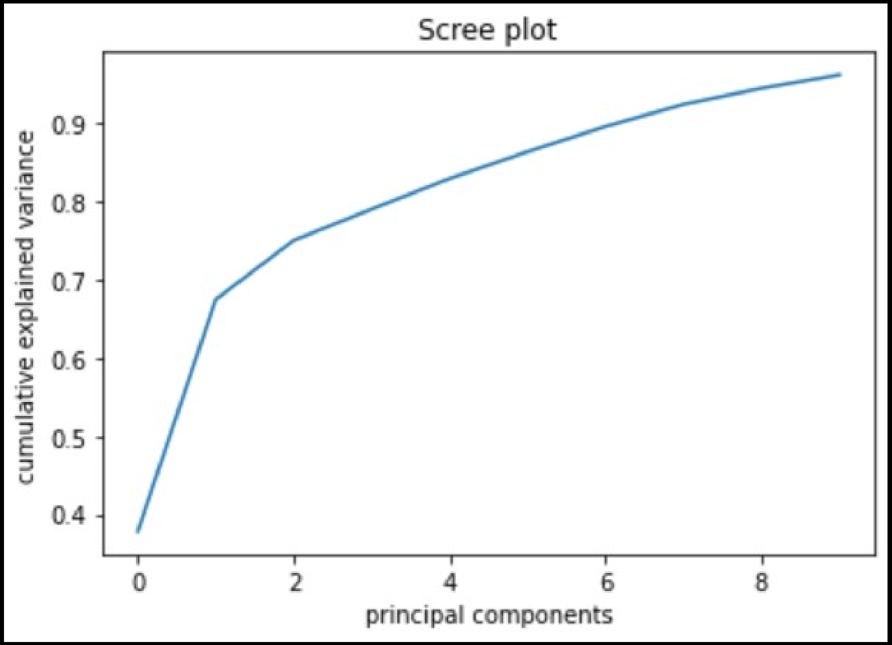
- Slight bump-up in weighted F1 score – 0.01.
- However ignored due to possible implications being too small

Saving time – optimising model

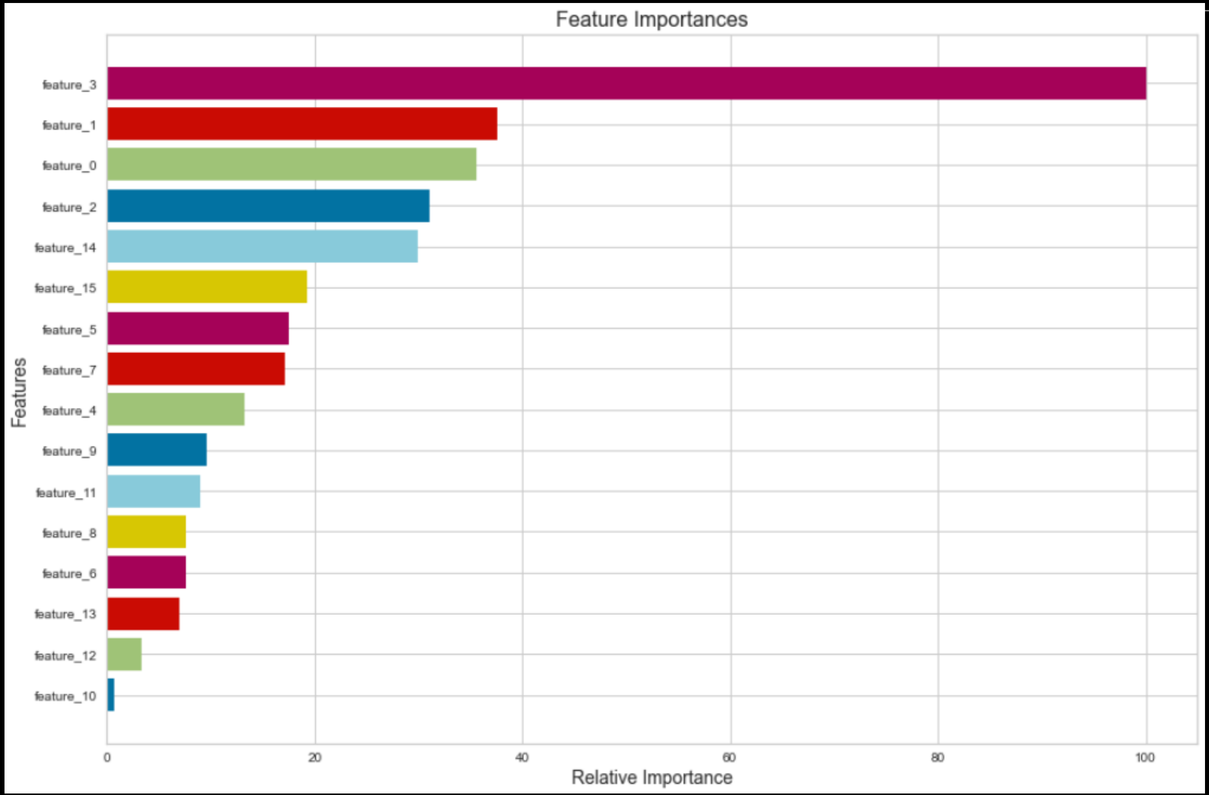
- Applying PCA
- Identifying feature importance manually

Applying PCA did not save time

- 10 components explained 95% of the variance in data
- However PCA application increased run time from 2.75 seconds to 7.10 seconds



Manual method with 6 features saved 0.22 seconds



Selected only feature_3, 1,0,2 & 14, using only top 5 features

Hyper Parameter tuning helped to reduce over-fitting on training set

- Basis multiple runs selected:
  - Criterion – Gini
  - Max-depth – 11
  - Max-features – Auto
  - n_estimators - 400

Model Evaluation

- Confusion Matrix
- AUC ROC Curve

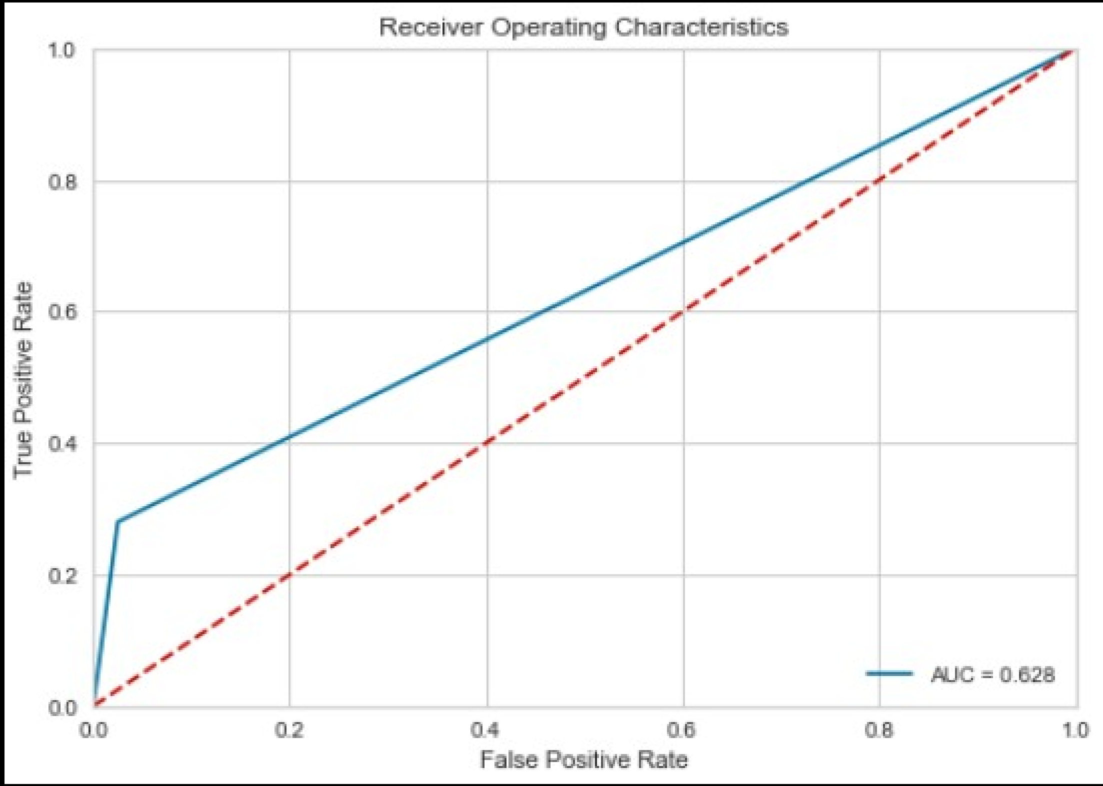
The confusion matrix

- Basis best hyper-tuned model

	predicted will not churn	predicted will churn
actual will not churn	5850	150
actual will churn	563	219

AUC / ROC curve

- The model can possibly be improved further
- Definitely better than pure random



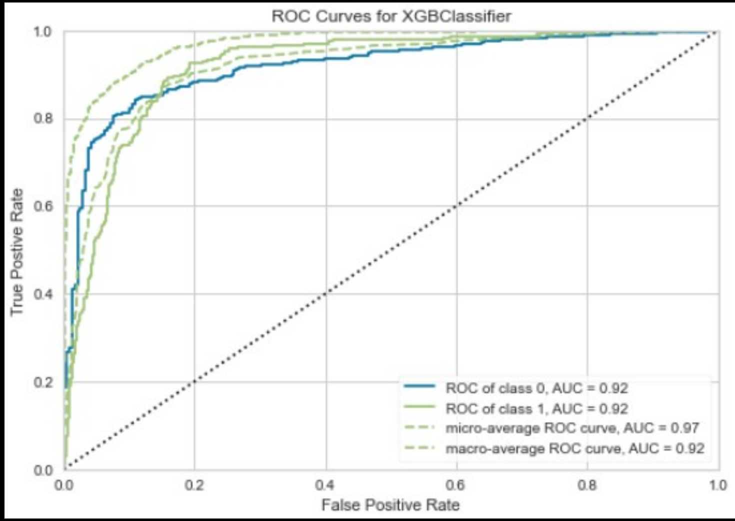
Fun with Auto ML

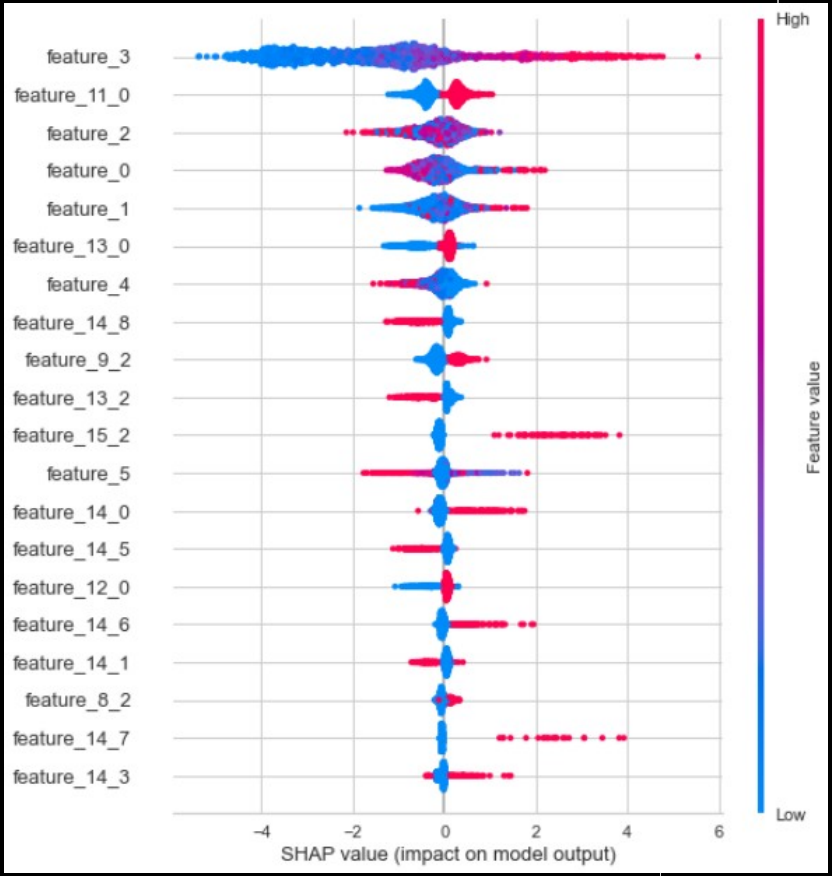
- Pycaret

- Tpot

Pycaret

- When optimised to F1 score:
- Recommended Extreme Gradient Boosting





Tpot

- Was able to provide a better score using Decision Tree Classifier – when hyper-tuned provided an improvement of the score to 0.559

Conclusion and way forward

- *It was extremely exciting working on this real world project from Kaggle. I was able to apply my learning across multiple parameters and identify what would possibly be currently the best model for this.*

- *Though spending more time on the model itself and additional computing power would definitely help yield a more hyper-tuned model.*