Advanced IBM Data Science:

PREDICTING TELCO CUSTOMER CHURN

Compiled by Sabelo Yalezo **Date:2020-08-14**

INTRODUCTION: USE CASE

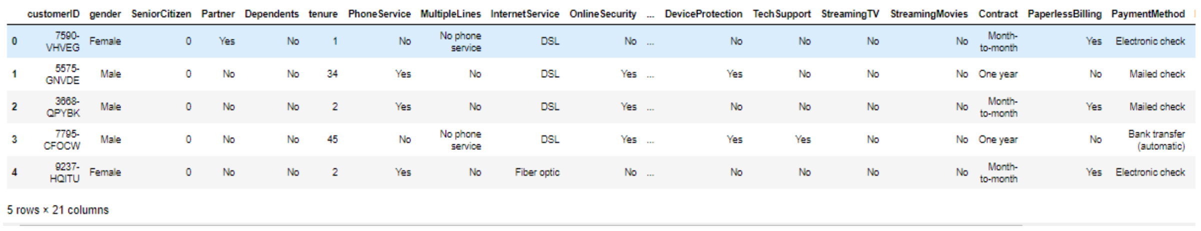
The telecom market in the SA is saturated and customer growth rates are low. They key focus of market players therefore is on retention and churn control. This project explores the churn dataset in Kaggle to identify the key drivers of churn and builds the best predictive model to predict churn. Customer churn, also known as customer attrition, occurs when customers stop doing business with a company or stop using a company's services. By being aware of and monitoring churn rate, companies are equipped to determine their customer retention success rates and identify strategies for improvement. We will use a machine learning model to understand the precise customer behaviours and attributes which signal the risk and timing of customer churn. In this project, I want to predict Telco customer churn based on their behaviour. Then question arise***: what the main contributing factors for Telecom customer churn are.***

DATA GATHERING AND PROCESSING

The data was sourced from Kaggle. This is public data; the enterprise data could not be obtained and publish without the proper process being followed. We categorised the data as follows:

- *Service categories* contains information about the service that are being utilized by customers.
- *Billing categories* contains the billing information.
- *Duration categories*, it's the duration of how long customers have been used services.
- *Support categories*, it indicates whether technical , security and backup services.
- *Personal category* determines the whether a customer is an old person or not, and whether the dependents, partner.

Telecon Churn Data: obtained data from https://www.kaggle.com/pavanraj159/telecom-customer-churn-prediction/data



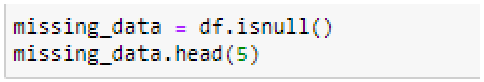
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month	Yes	Electronic check
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year	No	Mailed check
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month	Yes	Electronic check

5 rows × 21 columns

INITIAL DATA EXPLORATION

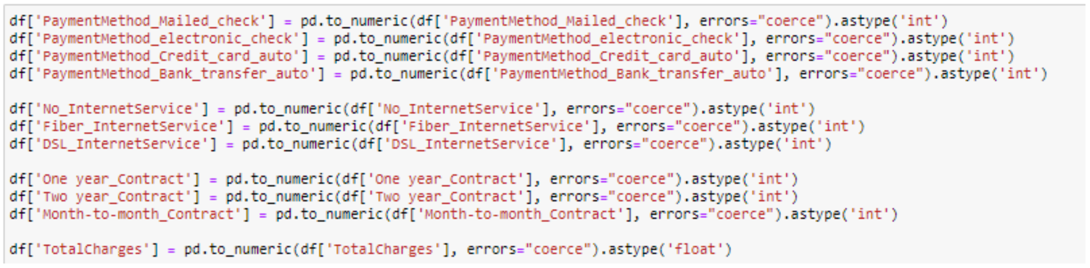
All the data wrangling was done in **Pandas** and **Numpy** library. Some techniques above might work better with some algorithms or datasets, while some of them might be beneficial in all cases. This article does not aim to go so much deep in this aspect. Tough, it is possible to write an article for every method above, I tried to keep the explanations brief and informative. I think the best way to achieve expertise in feature engineering is practicing different techniques on various datasets and observing their effect on model performances. We used the following techniques:

- **Evaluating for Missing Data**: The missing values are converted to Python's default. We use Python's built-in functions to identify these missing values. There are two methods to detect missing data: **.isnull() and .notnull().** The output is a boolean value indicating whether the value that is passed into the argument is in fact missing data.



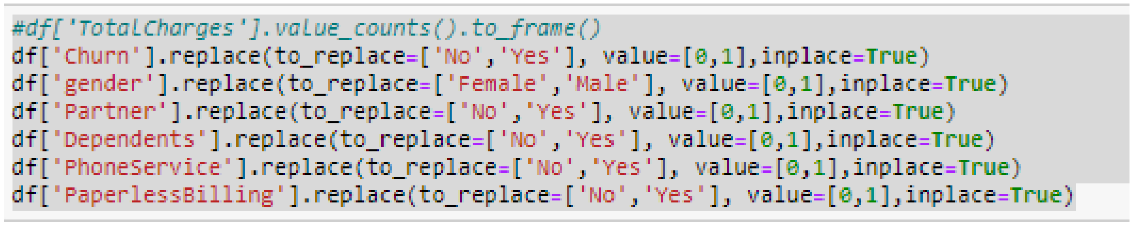
```
missing_data = df.isnull()
missing_data.head(5)
```

- **Alter Data Types** – Leveraging the correct data types assists in saving memory usage and can be a requirement for predictions to be performed against it.



```
df['PaymentMethod_Mailed_check'] = pd.to_numeric(df['PaymentMethod_Mailed_check'], errors="coerce").astype('int')
df['PaymentMethod_electronic_check'] = pd.to_numeric(df['PaymentMethod_electronic_check'], errors="coerce").astype('int')
df['PaymentMethod_Credit_card_auto'] = pd.to_numeric(df['PaymentMethod_Credit_card_auto'], errors="coerce").astype('int')
df['PaymentMethod_Bank_transfer_auto'] = pd.to_numeric(df['PaymentMethod_Bank_transfer_auto'], errors="coerce").astype('int')

df['No_InternetService'] = pd.to_numeric(df['No_InternetService'], errors="coerce").astype('int')
df['Fiber_InternetService'] = pd.to_numeric(df['Fiber_InternetService'], errors="coerce").astype('int')
df['DSL_InternetService'] = pd.to_numeric(df['DSL_InternetService'], errors="coerce").astype('int')

df['One year_Contract'] = pd.to_numeric(df['One year_Contract'], errors="coerce").astype('int')
df['Two year_Contract'] = pd.to_numeric(df['Two year_Contract'], errors="coerce").astype('int')
df['Month-to-month_Contract'] = pd.to_numeric(df['Month-to-month_Contract'], errors="coerce").astype('int')

df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors="coerce").astype('float')
```

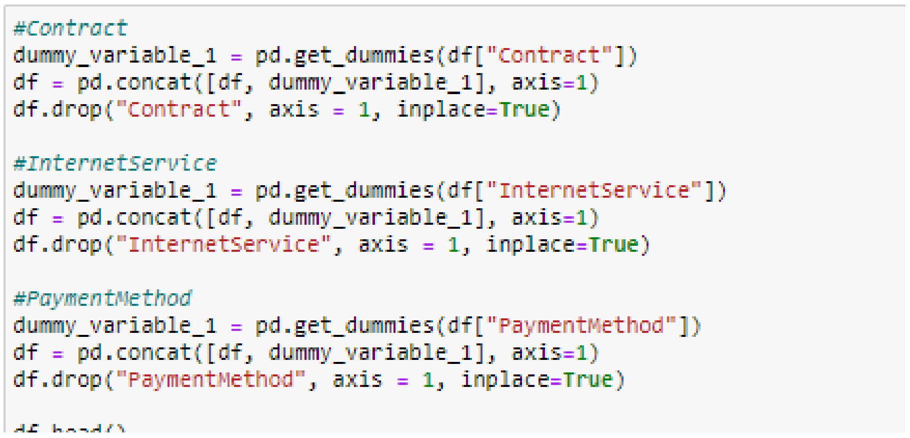
- We have dropped the observation were the "TotalPrice" was empty and the "CustomerID" feature since it was not needed. Moreover, we converted the categorical variables to numeric of simplified computation.



```
#df['TotalCharges'].value_counts().to_frame()
df['Churn'].replace(to_replace=['No','Yes'], value=[0,1],inplace=True)
df['gender'].replace(to_replace=['Female','Male'], value=[0,1],inplace=True)
df['Partner'].replace(to_replace=['No','Yes'], value=[0,1],inplace=True)
df['Dependents'].replace(to_replace=['No','Yes'], value=[0,1],inplace=True)
df['PhoneService'].replace(to_replace=['No','Yes'], value=[0,1],inplace=True)
df['PaperlessBilling'].replace(to_replace=['No','Yes'], value=[0,1],inplace=True)
```

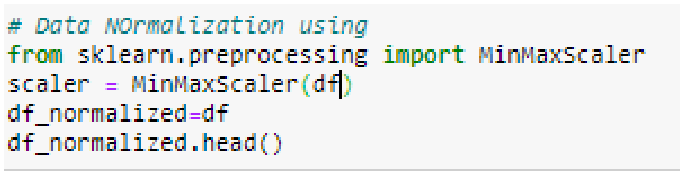
Feature Engineering

Feature selection is the process of choosing a subset of features, from a set of original features, based on a specific selection criterion. The main advantages of feature selection are: 1) reduction in the computational time of the algorithm, 2) improvement in predictive performance, 3) identification of relevant features, 4) improved data quality, and 5) saving resources in subsequent phases of data collection. This input data comprises features, which are usually in the form of structured columns. Algorithms require features with some specific characteristic to work properly. Here, the need for **feature engineering** arises. The included: Preparing the proper input dataset, compatible with the machine learning algorithm requirements. First, Improving the performance of machine learning models using **One-hot encoding**. It is one of the most common encoding methods in machine learning. This method spreads the values in a column to multiple flag columns and assigns **0** or **1** to them. These binary values express the relationship between grouped and encoded column.



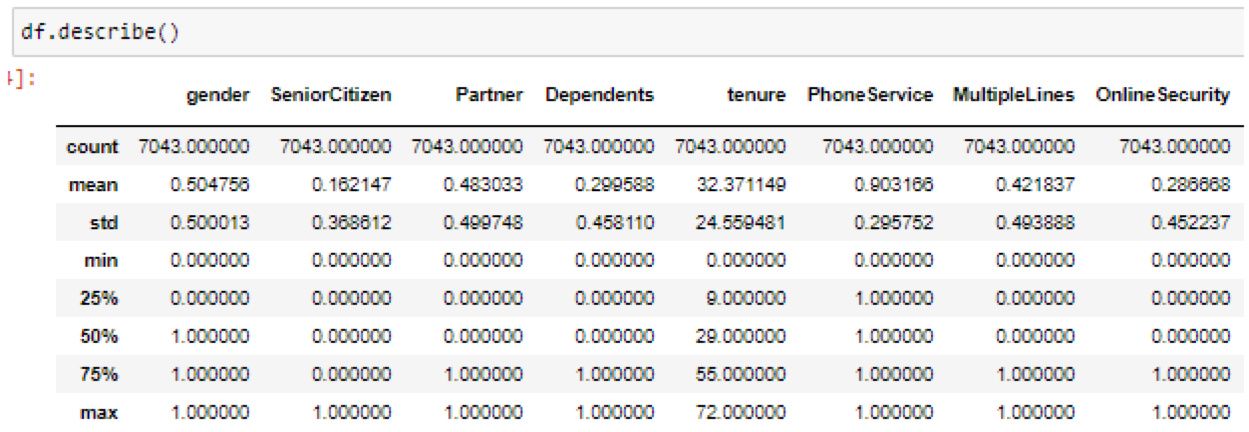
```
#Contract
dummy_variable_1 = pd.get_dummies(df["Contract"])
df = pd.concat([df, dummy_variable_1], axis=1)
df.drop("Contract", axis = 1, inplace=True)

#InternetService
dummy_variable_1 = pd.get_dummies(df["InternetService"])
df = pd.concat([df, dummy_variable_1], axis=1)
df.drop("InternetService", axis = 1, inplace=True)

#PaymentMethod
dummy_variable_1 = pd.get_dummies(df["PaymentMethod"])
df = pd.concat([df, dummy_variable_1], axis=1)
df.drop("PaymentMethod", axis = 1, inplace=True)

df.head()
```

Secondly, Scaling In most cases, the numerical features of the dataset do not have a certain range and they differ from each other. In real life, it is nonsense to expect age and income columns to have the same range. But from the machine learning point of view, how these two columns can be compared?



```
# Data NOrmalization using
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(df)
df_normalized=df
df_normalized.head()
```

Outlier Detection with Standard Deviation: If a value has a distance to the average higher than *x * standard deviation***,** it can be assumed as an outlier. So far, we have seen the Churn rate, but we are still unfamiliar with the data and its features, and what they are representing. What we must know is the distribution of data like variance, standard deviation, number of sample (count) or max min values. These type of information helps to understand the data, how normally distributed it is, or it has skewed distribution.

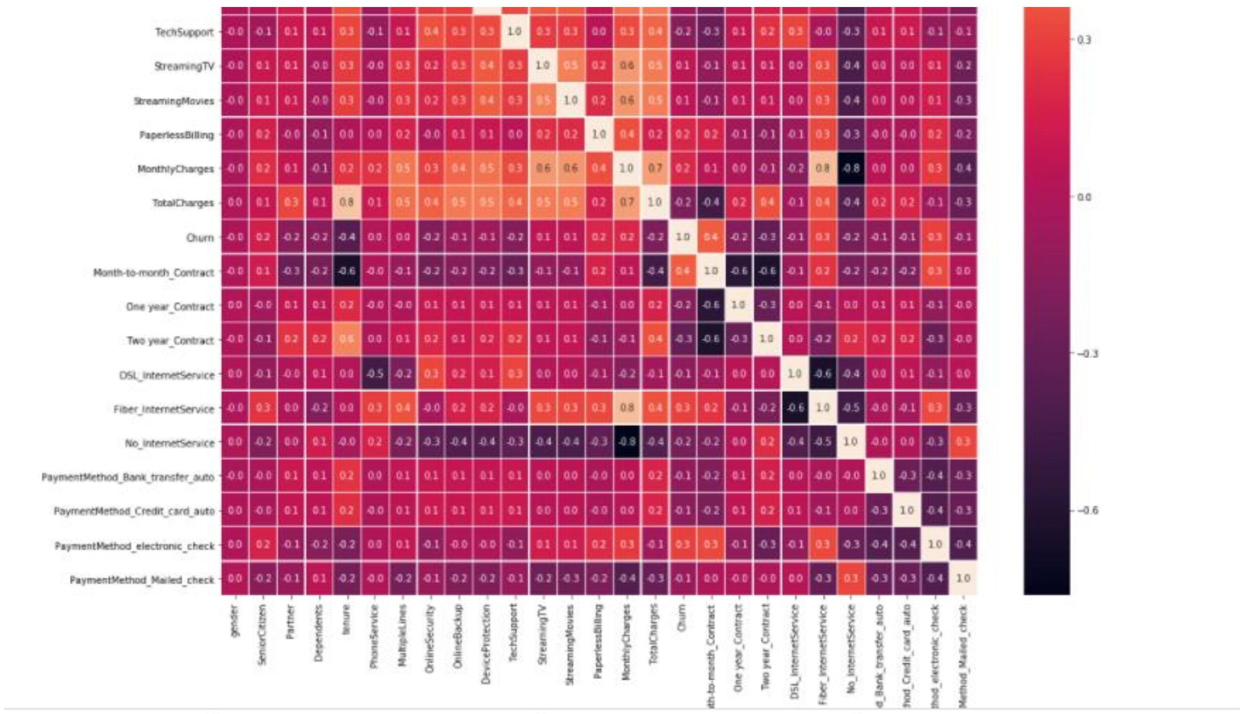


```
df.describe()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.504756	0.162147	0.483033	0.299588	32.371149	0.903166	0.421837	0.286668
std	0.500013	0.368612	0.499748	0.458110	24.559481	0.295752	0.493888	0.452237
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	9.000000	1.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	29.000000	1.000000	0.000000	0.000000
75%	1.000000	0.000000	1.000000	1.000000	55.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	72.000000	1.000000	1.000000	1.000000

FEATURE SELECTION

Let’s observe the correlation between all features and so we use this insight in feature selection for predictive model. After doing the EDA, we have used the correlation matrix to get the initial base model. We selected value based on the visualization we saw on correlation matrix.

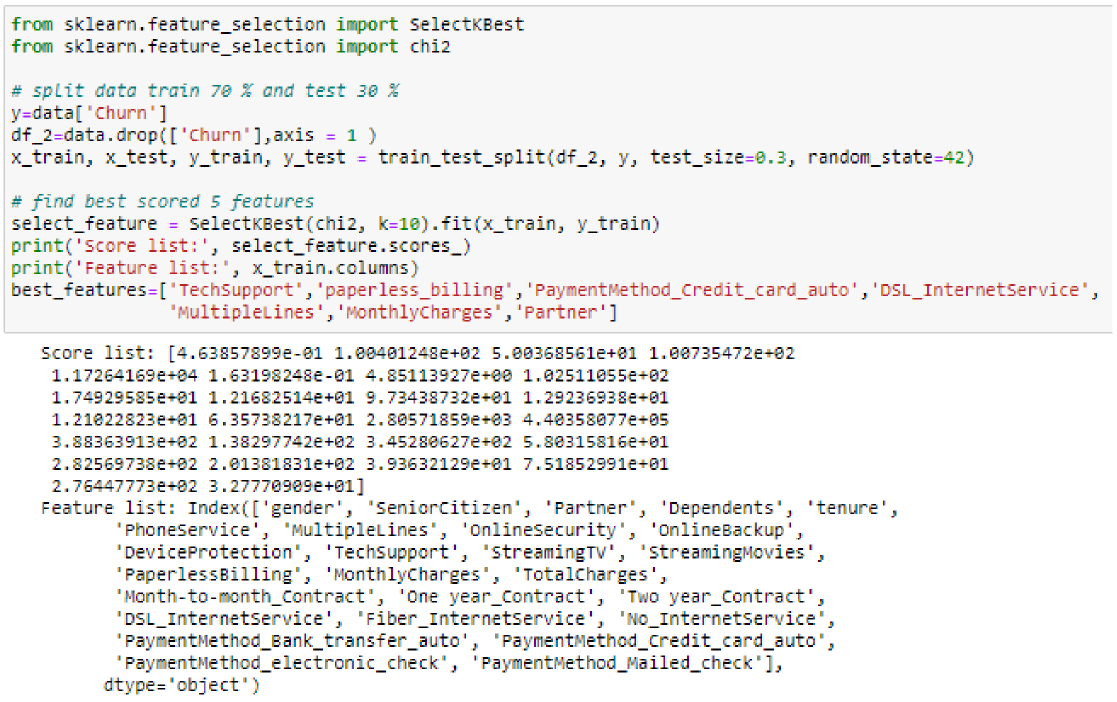


After the initial process modelling we realised that the method we used for selecting feature was not accurate. Hence we use the following 2 method to enhance the process of select features.

1. **Chi-Squared test.**

The chi-squared test determines whether there is a significant association between two categorical variables. We calculated the two-sided chi-squared p value to test the association between each categorical feature and the binary outcome at the 5% level of significance. We retained those features with a two-sided p<0.05.

UNIVARIATE SELECTION

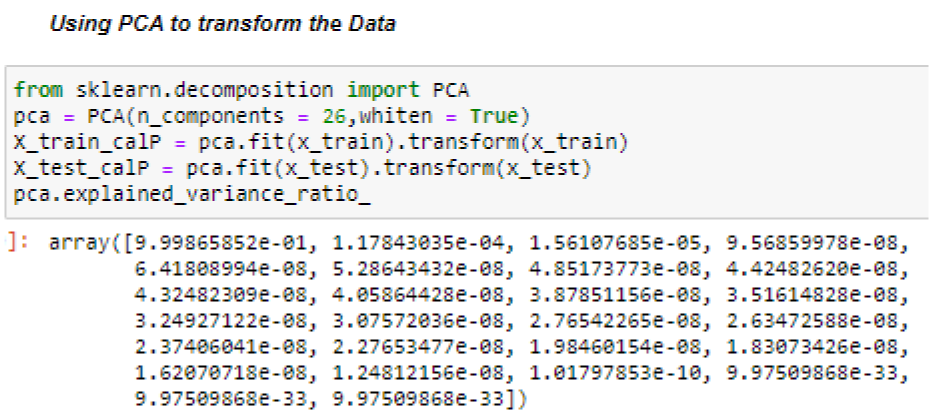


```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

# split data train 70 % and test 30 %
y=data['Churn']
df_2=data.drop(['Churn'],axis = 1 )
x_train, x_test, y_train, y_test = train_test_split(df_2, y, test_size=0.3, random_state=42)

# find best scored 5 features
select_feature = SelectKBest(chi2, k=10).fit(x_train, y_train)
print('Score list:', select_feature.scores_)
print('Feature list:', x_train.columns)
best_features=['TechSupport','paperless_billing','PaymentMethod_Credit_card_auto','DSL_InternetService',
               'MultipleLines','MonthlyCharges','Partner']
```

```
Score list: [4.63857899e-01 1.00401248e+02 5.00368561e+01 1.00735472e+02
 1.17264169e+04 1.63198248e-01 4.85113927e+00 1.02511055e+02
 1.74929585e+01 1.21682514e+01 9.73438732e+01 1.29236938e+01
 1.21022823e+01 6.35738217e+01 2.80571859e+03 4.40358077e+05
 3.88363913e+02 1.38297742e+02 3.45280627e+02 5.80315816e+01
 2.82569738e+02 2.01381831e+02 3.93632129e+01 7.51852991e+01
 2.76447773e+02 3.27770909e+01]
Feature list: Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'PaperlessBilling', 'MonthlyCharges', 'TotalCharges',
       'Month-to-month_Contract', 'One year_Contract', 'Two year_Contract',
       'DSL_InternetService', 'Fiber_InternetService', 'No_InternetService',
       'PaymentMethod_Bank_transfer_auto', 'PaymentMethod_Credit_card_auto',
       'PaymentMethod_electronic_check', 'PaymentMethod_Mailed_check'],
      dtype='object')
```

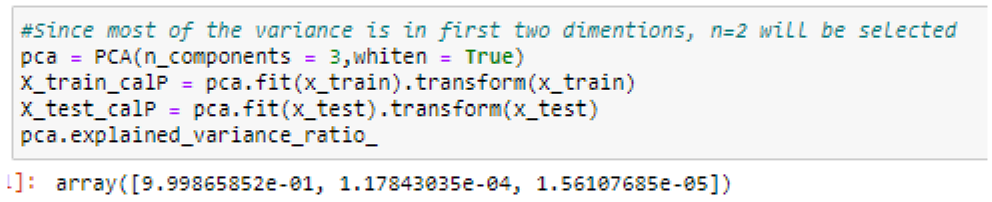
2. PCA

Large datasets are increasingly, to interpret such datasets, methods are required to drastically reduce their dimensionality in an interpretable way, such that most of the information in the data is preserved. In this project we use PCA to reduce the dimensionality of a dataset, while preserving as much statistical information as possible. This means that ‘preserving as much variability as possible’ translates into finding new variables that are linear functions of those in the original dataset, that successively maximize variance and that are uncorrelated with each other. Finding such new variables, the principal components (PCs), reduces to solving an eigenvalue/eigenvector problem. PCA can be based on either the covariance matrix or the correlation matrix. The choice between these analyses will be discussed. Although for inferential purposes a multivariate normal (Gaussian) distribution of the dataset is usually assumed, PCA as a descriptive tool needs no distributional assumptions and, as such, is very much an adaptive exploratory method which can be used on numerical data of various types. Because each eigenvalue is roughly the importance of its corresponding eigenvector, the proportion of variance explained is the sum of the eigenvalues of the features you kept divided by the sum of the eigenvalues of all features.


Using PCA to transform the Data

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 26,whiten = True)
X_train_calP = pca.fit(x_train).transform(x_train)
X_test_calP = pca.fit(x_test).transform(x_test)
pca.explained_variance_ratio_
```

```
]: array([9.99865852e-01, 1.17843035e-04, 1.56107685e-05, 9.56859978e-08,
          6.41808994e-08, 5.28643432e-08, 4.85173773e-08, 4.42482620e-08,
          4.32482309e-08, 4.05864428e-08, 3.87851156e-08, 3.51614828e-08,
          3.24927122e-08, 3.07572036e-08, 2.76542265e-08, 2.63472588e-08,
          2.37406041e-08, 2.27653477e-08, 1.98460154e-08, 1.83073426e-08,
          1.62070718e-08, 1.24812156e-08, 1.01797853e-10, 9.97509868e-33,
          9.97509868e-33, 9.97509868e-33])
```



```
#Since most of the variance is in first two dimentions, n=2 will be selected
pca = PCA(n_components = 3,whiten = True)
X_train_calP = pca.fit(x_train).transform(x_train)
X_test_calP = pca.fit(x_test).transform(x_test)
pca.explained_variance_ratio_
```

```
]: array([9.99865852e-01, 1.17843035e-04, 1.56107685e-05])
```

PROCESSING MODELLING

We have defined the models for **Random forest, Support Vector Machine, Decision Tree, KNN** and ***Gaussian***. While we were evaluating SVM after we applied the Univariate feature selection, our model got stuck for almost 50 minutes due intensive computing used by the model. Install Pyspark as we can see the computation using Pandas getting slow and the complexity of the model improves.

Another way to think of PySpark is a library that allows processing large amounts of data on a single machine or a cluster of machines. In a Python context, think of Pyspark has a way to handle parallel processing without the need for the threading or multiprocessing modules. All the complicated communication and synchronization between threads, processes, and even different CPUs is handled by Spark. After that we were able to run our model easily and efficiently. We were then able to create _KERAs and k-Means algorithms.

MODEL EVALUATION METRICS

We use Confusion matrix to evaluate the validity and accuracy of our model. A confusion matrix is an n*n matrix, where N is the number of classes in dependent variable or target. Majority of the time, we have n=2, and hence we get a 2*2 matrix. Now let’s talk about the statistics we can measure from this confusion matrix.

Confusion Matrix Target	
True Positive (A)	**True Positive (B)**
False Negative (C)	**True Negative (B)**

1. Accuracy: (A+D)/ (A+B+C+D)
2. Misclassification error: (B+C) / (A+B+C+D)
3. Positive Predictive Value or Precision: A / (A+B)
4. Negative Predictive Value: D / (C+D)
5. Sensitivity or Recall or hit rate or true positive rate: A / (A+C)
6. Specificity or true negative rate: D / (B+D)

Different parameters have been used in different business objectives and domains. We are working on a churn model where you need to cover as much as possible then you will focus on precision. There is one more measure that is dependent upon precision and recall. F1-Score is the harmonic mean of precision and recall values for a classification problem. F1 Score formula is:

$$2^* \frac{(\text{Precision} \; ^* \; \text{Recall})}{(\text{Precision} + \text{Recall})}$$

Below is the table that summarizes the model evaluation.

	Algorithm	F1 scores	Accuracy	Precision	Recall

Supervised learning	Random forest (PCA)	0.45	0.74	0.53	0.39
	Support Vector(PCA)	0.48	0.78	0.65	0.37
	KNN (PCA)	0.48	0.75	0.55	0.44
	Linear regression (PCA)	0.50	0.77	0.61	0.42
	Decision Tree (PCA)	0.48	0.71	0.46	0.50
	Gaussian (PCA)	0.53	0.74	0.54	0.52
Deep Learning	Keras Model	-	0.69	-	-
Unsupervised Learning	K-Means	-	0.97	-	-

Recommendation and Insights

The model can help with the following:

- Identify root causes for churn intent. Then take steps to address those concerns.
- Proactively check on customers to confirm that their issue is fixed.
- Identify root causes for churn intent and take steps to alleviate those concerns.

From the current dataset we see that the current churn rate is 26%, Telecom companies need to rerun the model on monthly to check likelihood of their customers leaving their network. This will help to mitigate root cause analysis. In the project we found that the following feature has got a greet impact on Churn. In other the significant features for the churn model using the current dataset are as follows:

- TechSupport,
- Paperless_billing,
- PaymentMethod_Credit_card_auto,
- DSL_InternetService,
- MultipleLines,
- MonthlyCharges,
- Partner

The customers that do not receive technical support from their ISPs are likely to churn. Moreover, since DSL is being overtaken by Fibre in the recent years, DSL services have impact of churn rate.

IN future we would like to get more data, like data with more than 1Million observation. We believe this data we mimic the real-world example and would results in better model that can predict churn with high accuracy.