Customer Churn Project

Introduction :

Customer churn analysis refers to the customer attrition rate in a company. This analysis helps SaaS companies identify the cause of the churn and implement effective strategies for retention. Gathers available customer behavior, transactions, demographics data and usage pattern.
Churn analysis is the evaluation of a company’s customer loss rate in order to reduce it. Also referred to as customer attrition rate, churn can be minimized by assessing your product and how people use it.
Customer churn (also known as customer attrition) refers to when a customer (player, subscriber, user, etc.) ceases his or her relationship with a company. Online businesses typically treat a customer as churned once a particular amount of time has elapsed since the customer’s last interaction with the site or service. The full cost of churn includes both lost revenue and the marketing costs involved with replacing those customers with new ones. Reducing churn is a key business goal of every online business.

Defining the problem:

Customer churn is when a company’s customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher volume of service consumption and can generate additional customer referrals. Customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them. The vast volumes of data collected about customers can be used to build churn prediction models. Knowing who is most likely to defect means that a company can priorities focused marketing efforts on that subset of their customer base.
Preventing customer churn is critically important to the telecommunications sector, as the barriers to entry for switching services are so low.

Data Pre-processing :

The purpose of preprocessing is to convert raw data into a form that fits machine learning. Structured and clean data allows a data scientist to get more precise results from an applied machine learning model. The technique includes data formatting, cleaning, and sampling.

- **Hardware and Software Requirements and Tools Used**

- *Tools: Python 3.8.5, Jupyter Notebook, Numpy, Pandas, Matplotlib, Seaborn, Scikit-learn, Scipy*
- *Techniques: Logistic Regression, SVC, Random Forest Classifier, Ada Boost Classifier, Decision Tree Classifier, Gradient Boosting Classifier, KNeighborsClassifier*
- *Hardware: I3 processor, 4GB RAM*

- **Model/s Development and Evaluation :**

  I have used the below models for classification:

- LogisticRegression
- SVC
- RandomForestClassifier
- AdaBoostClassifier
- DecisionTreeClassifier
- GradientBoostingClassifier
- KNeighborsClassifier

- Identification of possible problem-solving approaches (methods)

- Read the data (from csv)
- Identify the dependent and independent variables.
- Check if the data has missing values or the data is categorical or not.
- If yes, apply basic data preprocessing operations to bring the data in a go to go format.
- Now split the data into the groups of training and testing for the respective purpose.
- After splitting data, fit it to a most suitable model. (How to find a suitable model is answered below)
- Validate the model. If satisfactory, then go with it, else tune the parameters and keep testing. In a few cases, you can also try different algorithms for the same problem to understand the difference between the accuracies.
- From step 7 one can also learn about accuracy paradox.
- Visualize the data.

Data Analysis:

Data analysis is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making.Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains.[2] In today's business world, data analysis plays a role in making decisions more scientific and helping businesses operate more effectively.

Procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data.

There are several phases that can be distinguished, described below. The phases are iterative, in that feedback from later phases may result in additional work in earlier phases.

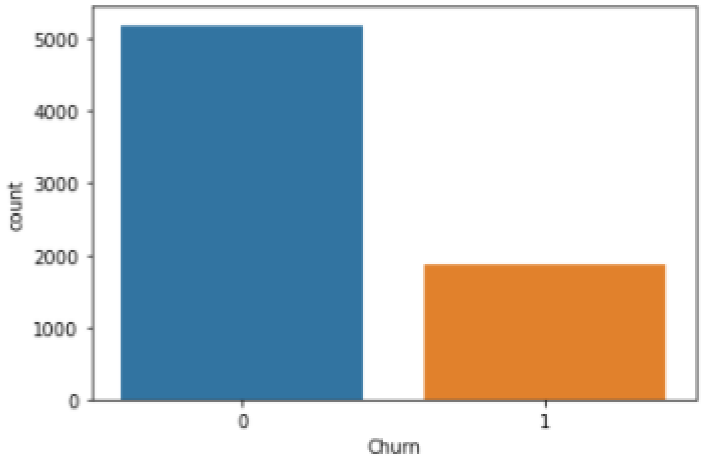
EDA :

Once the datasets are cleaned, it can then be analyzed. Analysts may apply a variety of techniques, referred to as exploratory data analysis, to begin understanding the messages contained within the obtained data. The process of data exploration may result in additional data cleaning or additional requests for data; thus, the initialization of the *iterative phases* mentioned in the lead paragraph of this section. Descriptive statistics, such as, the average or median, can be generated to aid in understanding the data. Data visualization is also a technique used, in which the analyst is able to examine the data in a graphical format in order to obtain additional insights, regarding the messages within the data.

Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis (IDA), which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed.

```
sns.countplot(df["Churn"])
```

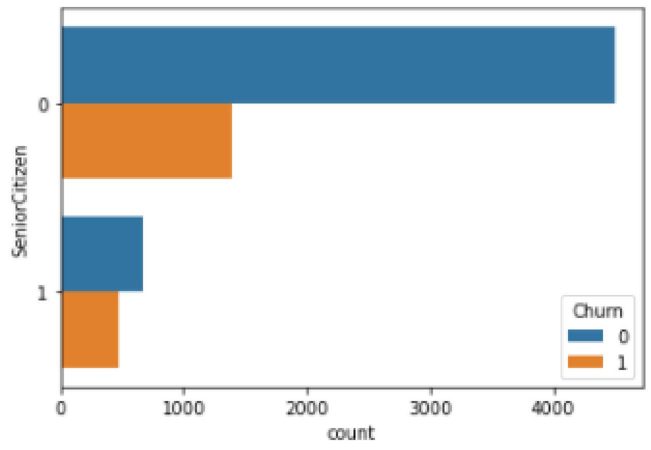
```
<AxesSubplot:xlabel='Churn', ylabel='count'>
```



There is class imbalance issue exists.

```
sns.countplot(y='SeniorCitizen', hue='Churn', data = df)
```

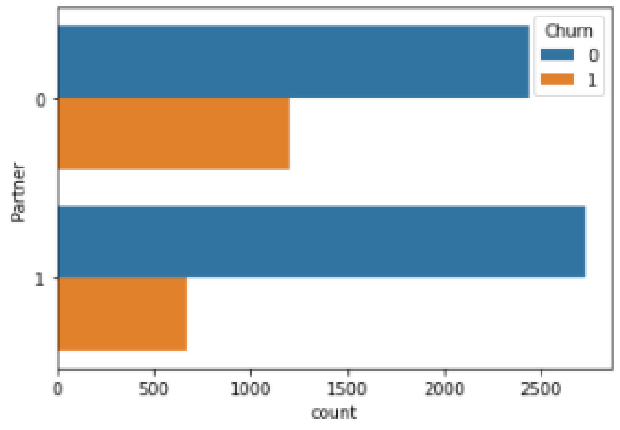
```
<AxesSubplot:xlabel='count', ylabel='SeniorCitizen'>
```



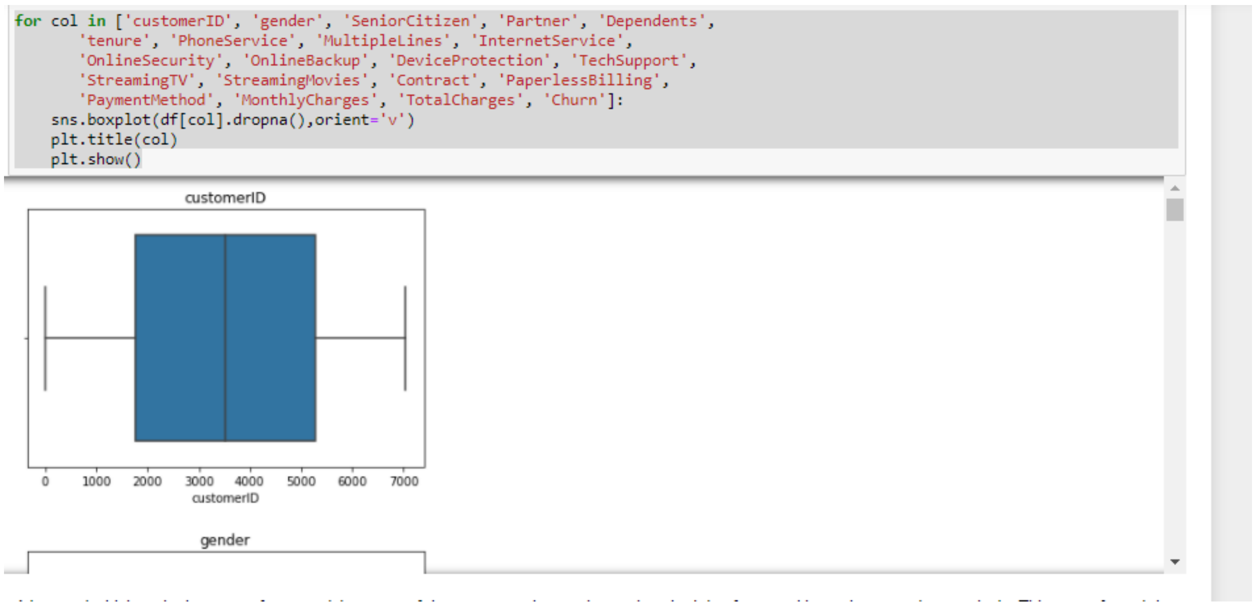
In Customer_Churn dataset has no missing values.In this plot is showing SeniorCitizen of the dataset.

```
sns.countplot(y='Partner', hue='Churn', data = df)
```

```
<AxesSubplot:xlabel='count', ylabel='Partner'>
```



In Customer_Churn dataset has no missing values.In this plot is showing Partner of the dataset.

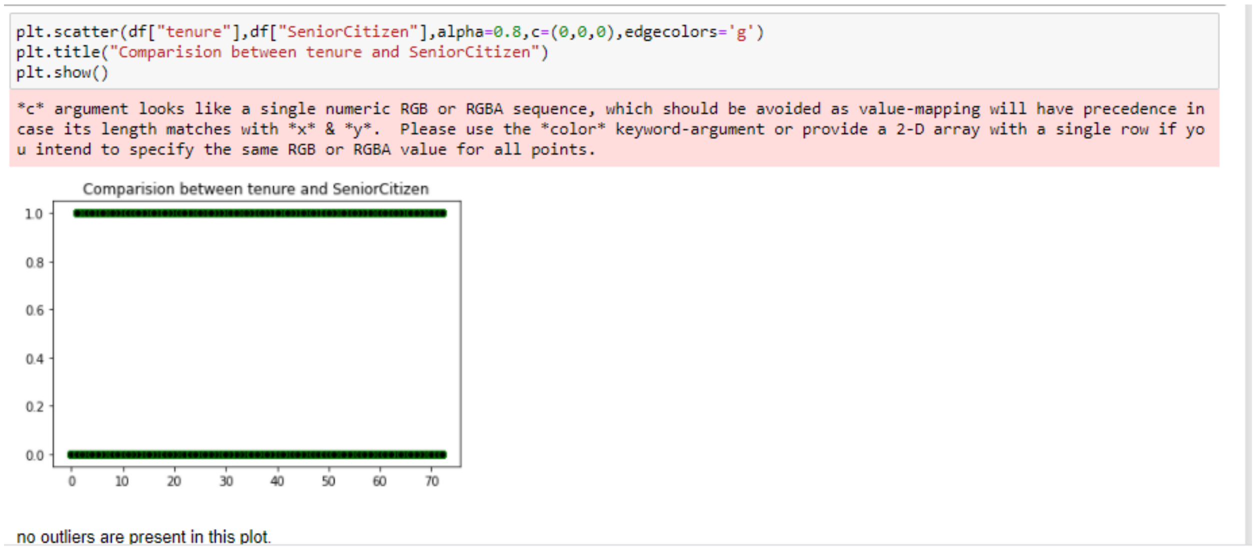


```
for col in ['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn']:
    sns.boxplot(df[col].dropna(),orient='v')
    plt.title(col)
    plt.show()
```

Bivariate analysis

Comparing multiple variables simultaneously is also another useful way to understand your data. When you have two continuous variables, a scatter plot is usually used. You can use a boxplot to compare one continuous and one categorical variable.

Scatter plots primary uses are to observe and show relationships between two numeric variables. The dots in a scatter plot not only report the values of individual data points, but also patterns when the data are taken as a whole. A scatter plot can also be useful for identifying other patterns in data.



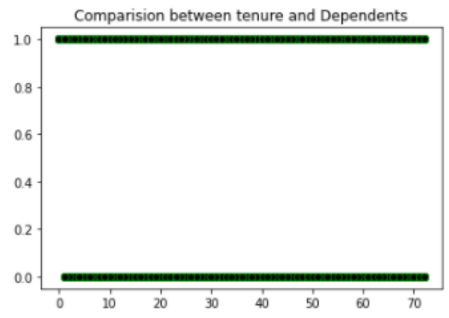
```
plt.scatter(df["tenure"],df["SeniorCitizen"],alpha=0.8,c=(0,0,0),edgecolors='g')
plt.title("Comparision between tenure and SeniorCitizen")
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```

no outliers are present in this plot.

```
plt.scatter(df["tenure"],df["Dependents"],alpha=0.8,c=(0,0,0),edgecolors='g')
plt.title("Comparision between tenure and Dependents")
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```



No outliers are present.

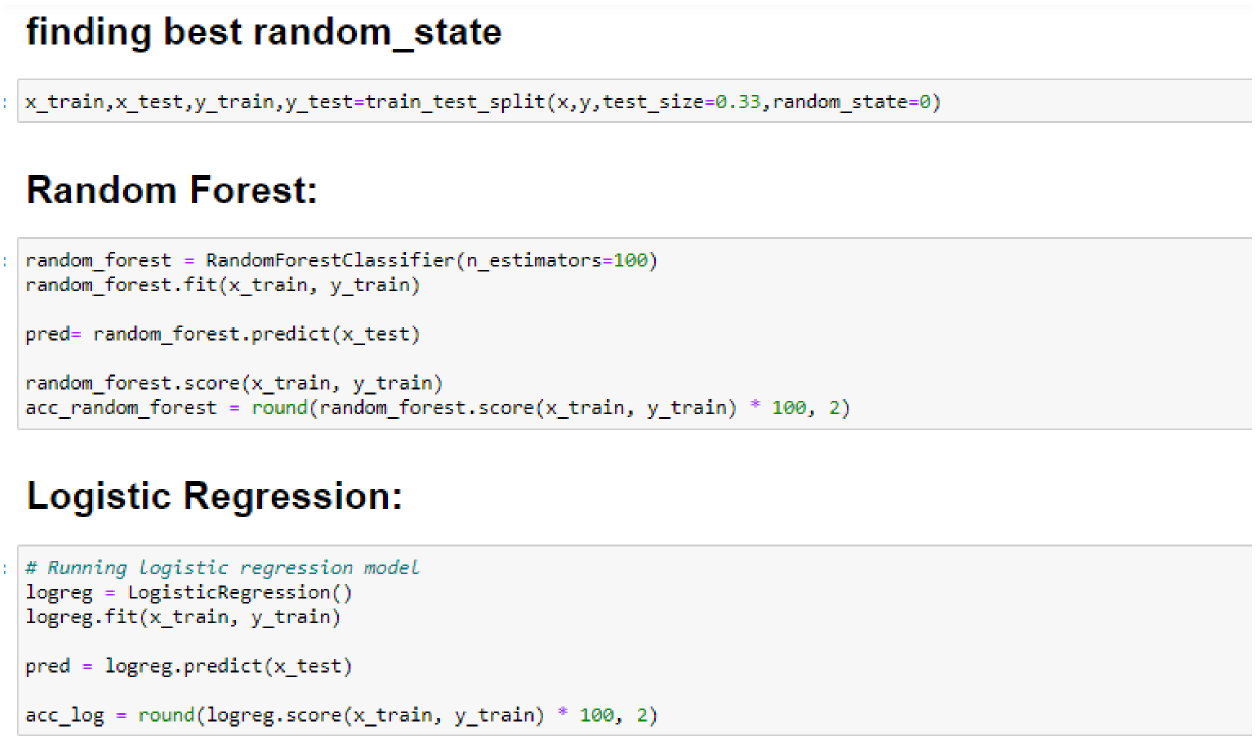
Building Machine Learning Model:

A **machine learning model** is built by **learning** and generalizing from training data, then applying that acquired knowledge to new data it has never seen before to make predictions and fulfill its purpose. Lack of data will prevent you from **building** the **model**, and access to data isn't enough.

The **model building** process involves setting up ways of collecting **data**, understanding and paying attention to what is important in the **data** to answer the questions you are asking, finding a statistical, mathematical or a simulation **model** to gain understanding and make predictions.

A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data.

Once you have trained the model, you can use it to reason over data that it hasn't seen before, and make predictions about those data. For example, let's say you want to build an application that can recognize a user's emotions based on their facial expressions. You can train a model by providing it with images of faces that are each tagged with a certain emotion, and then you can use that model in an application that can recognize any user's emotion.



finding best random_state

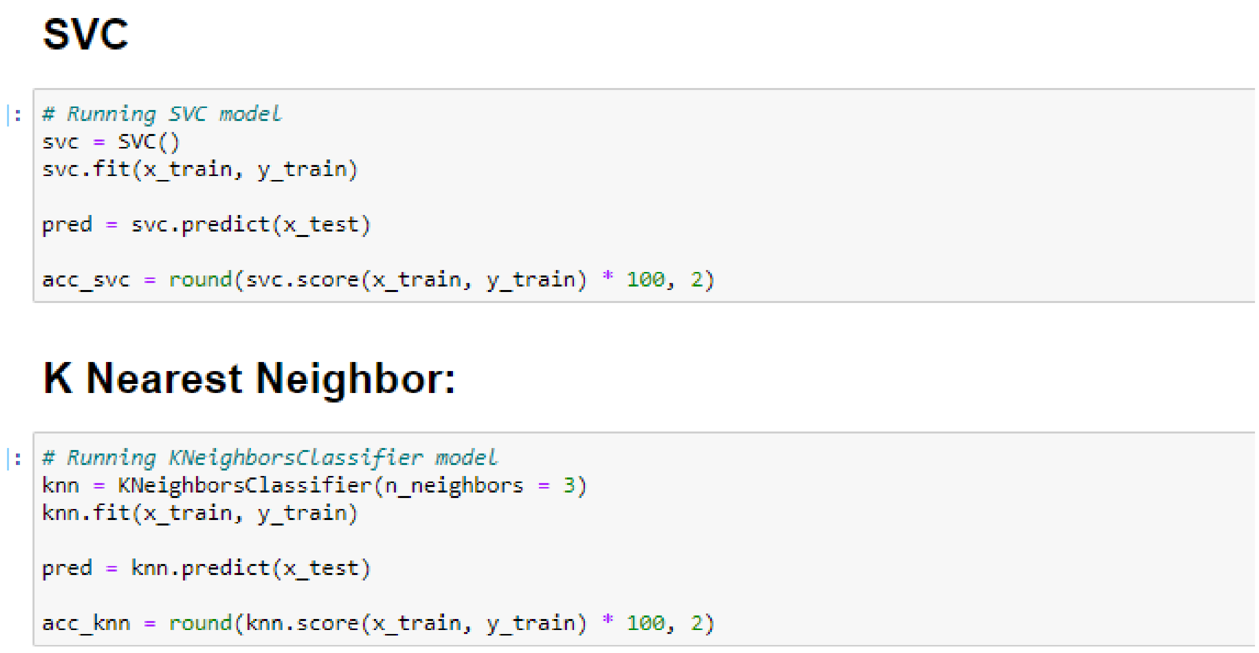
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=0)
```

Random Forest:

```
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(x_train, y_train)

pred= random_forest.predict(x_test)

random_forest.score(x_train, y_train)
acc_random_forest = round(random_forest.score(x_train, y_train) * 100, 2)
```

Logistic Regression:

```
# Running logistic regression model
logreg = LogisticRegression()
logreg.fit(x_train, y_train)

pred = logreg.predict(x_test)

acc_log = round(logreg.score(x_train, y_train) * 100, 2)
```



SVC

```
# Running SVC model
svc = SVC()
svc.fit(x_train, y_train)

pred = svc.predict(x_test)

acc_svc = round(svc.score(x_train, y_train) * 100, 2)
```

K Nearest Neighbor:

```
# Running KNeighborsClassifier model
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train, y_train)

pred = knn.predict(x_test)

acc_knn = round(knn.score(x_train, y_train) * 100, 2)
```

AdaBoostClassifier:

```
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)

pred = ad.predict(x_test)

acc_ad = round(ad.score(x_train, y_train) * 100, 2)
```

Decision Tree:

```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(x_train, y_train)

pred = decision_tree.predict(x_test)

acc_decision_tree = round(decision_tree.score(x_train, y_train) * 100, 2)
```

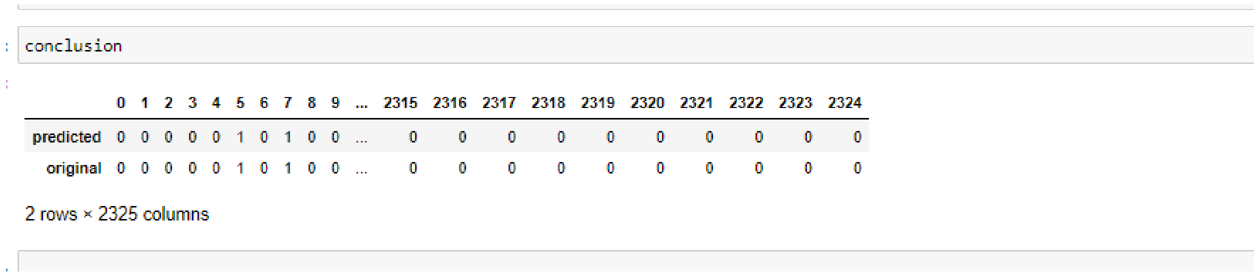
GradientBoostingClassifier model:

```
gbc=GradientBoostingClassifier()
gbc.fit(x_train,y_train)

pred = gbc.predict(x_test)

acc_gbc = round(gbc.score(x_train, y_train) * 100, 2)
```

- **Conclusion:**

Another great quality of Ada boost is that they make it very easy to measure the relative importance of each feature. Sklearn measure a features importance by looking at how much the treee nodes, that use that feature, reduce impurity on average (across all trees in the forest). It computes this score automatically for each feature after training and scales the results so that the sum of all importance is equal to 1.

The best score in this dataset is SVC model. The best score is 80% in this model.



```
conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324
predicted	0	0	0	0	0	1	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
original	0	0	0	0	0	1	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0

2 rows × 2325 columns