A COMPLETE DATA SCIENCE MODEL

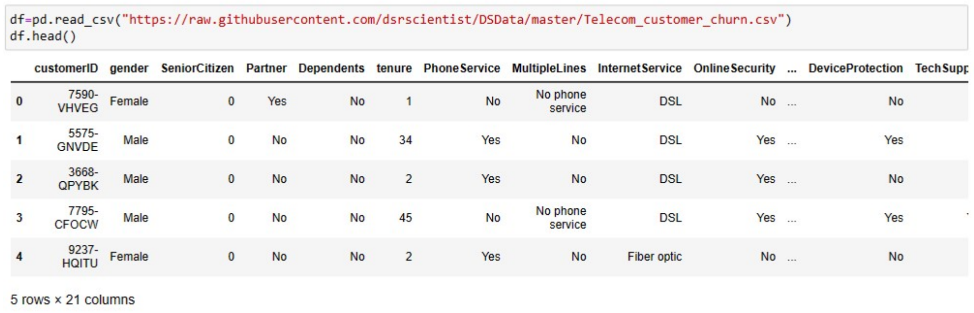
1. <u>PROBLEM STATEMENT</u>: Customer churn is when a company's customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, 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 prioritise 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.

   You will examine customer data from IBM Sample Data Sets with the aim of building and comparing several customer churn prediction models.

2. <u>DATA ANALYSIS</u>**:**



```
df=pd.read_csv("https://raw.githubusercontent.com/dsrscientist/DSData/master/Telecom_customer_churn.csv")
df.head()
```

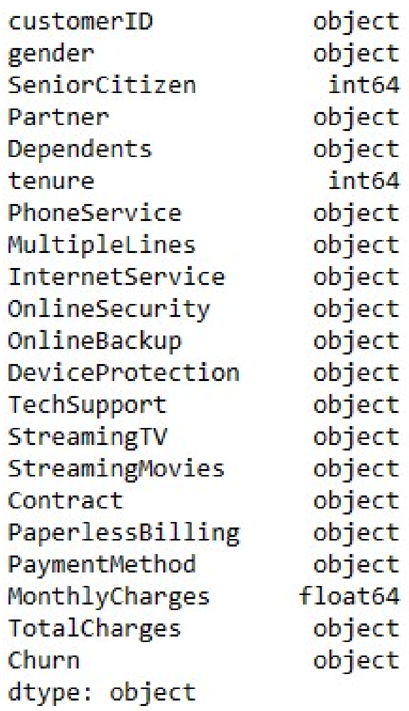
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows × 21 columns

The given example of dataset is a structured data and the aim is to build and compare several customer churn prediction models which is a regression model with TotalCharges as the target variable,

The columns of the dataset are: 'customerID', 'gender', 'SeniorCitizen', ' Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'

3. EDA:
   - Checking for the data types using:
     df.dtypes

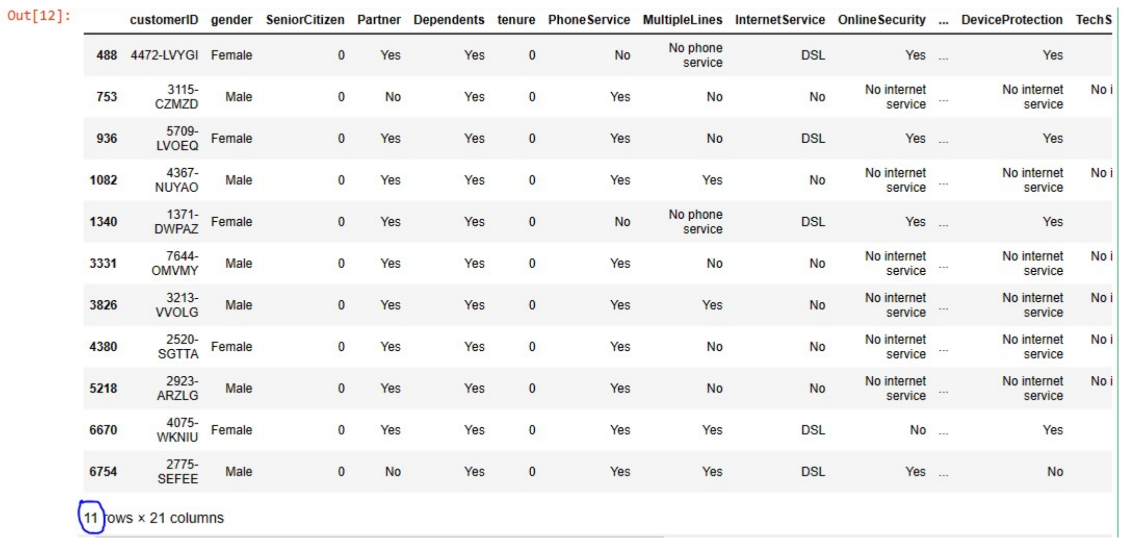


```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

     these are the data types of all the columns , since this is a regression model and the object type of the target variable is object where in it should be actually float for the machine to understand we will try to change it to float by the following step:
     df["TotalCharges"]=df["TotalCharges"].astype(float)
     df.TotalCharges.dtypes

   - Checking for the null values using:
     df.isnull().sum()
     Which gives the result 0 , which means there are no null values

   - Checking for space or blank data in the dataset using:
     df.loc[df['TotalCharges']==" "]



Out[12]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechS
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	
753	3115-CZMZD	Male	0	No	Yes	0	Yes	No	No	No internet service	...	No internet service	No i
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	...	Yes	
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No i
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No i
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No i
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No i
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No i
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	...	Yes	
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	...	No	

11 rows × 21 columns

Which means there are 11 rows where void spaces are present, hence we will replace it with nan values and the remove those null values using:

```
df["TotalCharges"]=df["TotalCharges"].replace(" ",np.nan)
```

```
df.isnull().sum()
```

```
df["TotalCharges"]=df["TotalCharges"].fillna(np.mean(df["TotalCharges"]))
```

```
df.isnull().sum()
```
(checking again for the cleared null values

- Visualization of data: passing values to feature_visualization

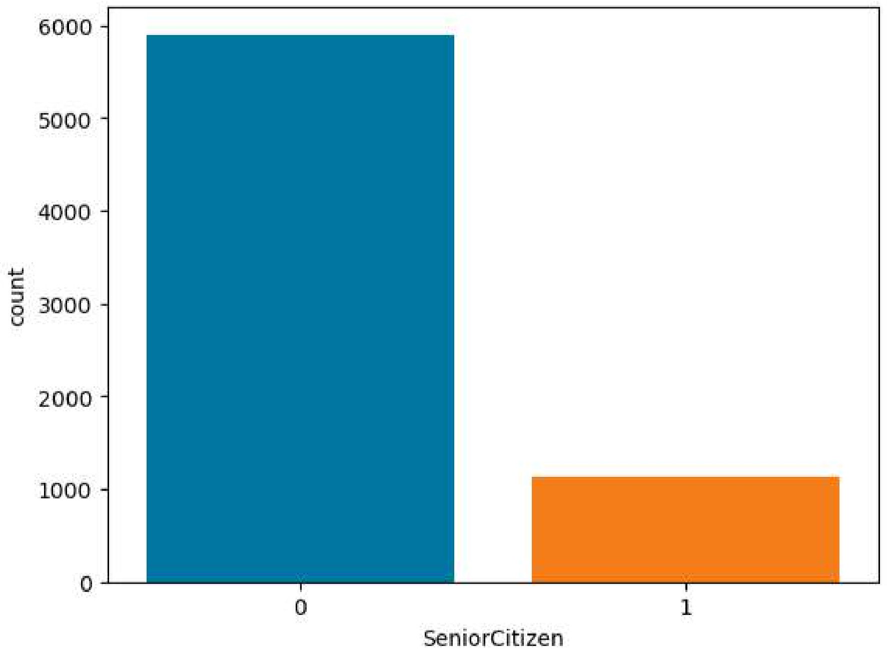
  ```
  feature_visualization=df[['gender','SeniorCitizen','Partner','Dependents','PhoneService','MultipleLines','InternetService','OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies','Contract','PaperlessBilling','PaymentMethod','Churn']].copy()
  ```

  then checking for all distribution and relationship between each features with the targets using:

  ```
  import seaborn as sns

  ax=sns.countplot(x="gender",data=feature_visualization)
  print(feature_visualization["gender"].value_counts())
  ```

```
ax=sns.countplot(x="SeniorCitizen",data=feature_visualization)
print(feature_visualization["SeniorCitizen"].value_counts())
```

```
0    5901
1    1142
Name: SeniorCitizen, dtype: int64
```



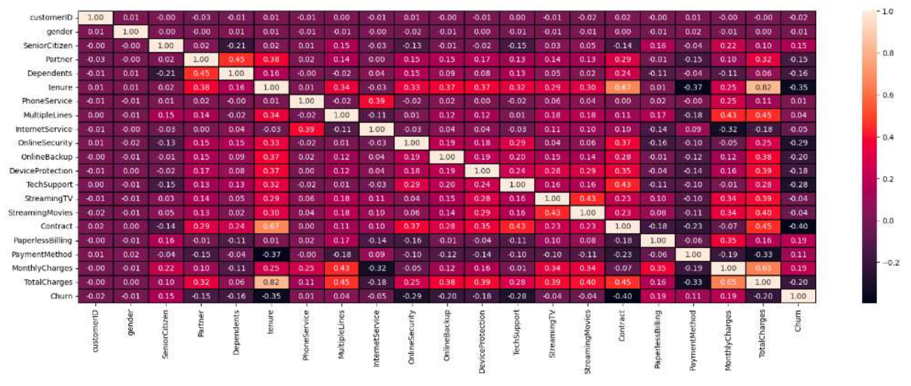
Continuing this steps for all the features

- Checking correlation with the target columns :
  df.corr()['TotalCharges'].sort_values()

  plt.figure(figsize=(22,7))
  sns.heatmap(df.corr(),annot=True,linewidth=0.1,linecolor="black",fmt="0.2f")

```
plt.figure(figsize=(22,7))
sns.heatmap(df.corr(),annot=True,linewidth=0.1,linecolor="black",fmt="0.2f")
```

```
<AxesSubplot:>
```



Here tenure and totalcharges are highly correlated, contract and tenure is highly correlated, likewise monthly charges and total charges

plt.figure(figsize=(22,7))

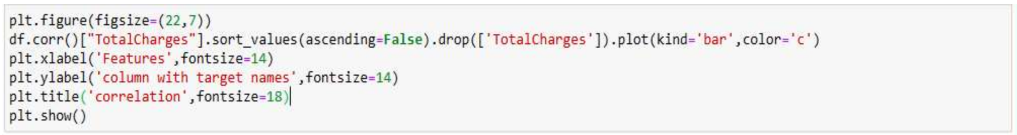
df.corr()["TotalCharges"].sort_values(ascending=False).drop(['TotalCharges']).plot(kind='bar',color='c')

plt.xlabel('Features',fontsize=14)

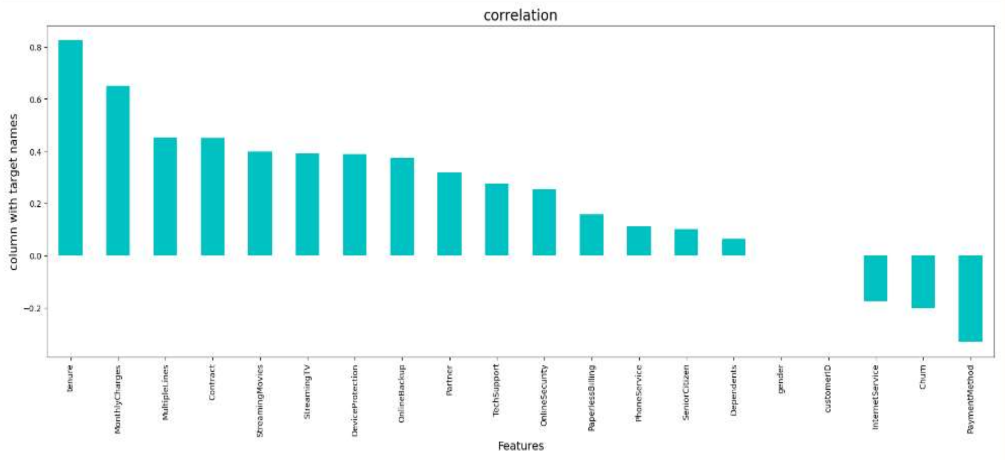
plt.ylabel('column with target names',fontsize=14)

plt.title('correlation',fontsize=18)

plt.show()



```
plt.figure(figsize=(22,7))
df.corr()["TotalCharges"].sort_values(ascending=False).drop(['TotalCharges']).plot(kind='bar',color='c')
plt.xlabel('Features',fontsize=14)
plt.ylabel('column with target names',fontsize=14)
plt.title('correlation',fontsize=18)
plt.show()
```



We can find out the correlation with the target variable through the barplot

- Checking for the skewness:
  df.skew()

  keeping +/-0.5 as threshold there is now skewness

    1) Senior citizen: categorical
    2) Dependents: categorical
    3) Phone service: categorical
    4) Contract: categorical
    5) Totalcharges: target variable
    6) Churn: categorical

- Checking for outliers :

```
df['SeniorCitizen'].plot.box()
df['TotalCharges'].plot.box()
df['MonthlyCharges'].plot.box()
```

our data is becoming biased as it not considering senior citizen hence we will not remove outliers

- Encoding :

```
from sklearn.preprocessing import OrdinalEncoder
enc=OrdinalEncoder()
for i in df.columns:
   if df[i].dtypes=="object":
      df[i]=enc.fit_transform(df[i].values.reshape(-1,1))
```

4.BUILDING MACHINE LEARNING MODELS:

- Separating the columns into features and targets

```
features=df.drop("TotalCharges",axis=1)
target=df['TotalCharges']
```

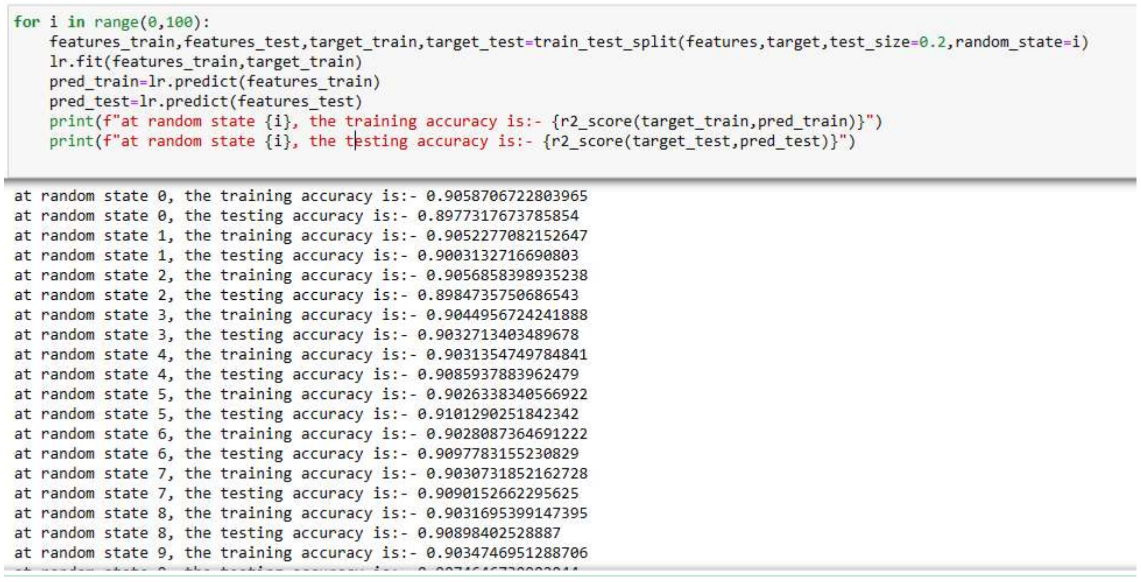
- Scaling the data using min-max scaler:importing different kind of libraries , we will not apply the scaling methods on categorical column and target variable but we will apply on continuous data if required

```
from sklearn.preprocessing import MinMaxScaler
nms=MinMaxScaler()
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

dividing the data into two parts (test data and train data using train train split)

and making prediction on train data and test data by using model name called predict method

we are checking the accuracy and calculating the r2 score, r2 score pulls out the difference between the training , testing score with the predicted one , from the range from 0, 99 it gives the score at random states



```
for i in range(0,100):
    features_train,features_test,target_train,target_test=train_test_split(features,target,test_size=0.2,random_state=i)
    lr.fit(features_train,target_train)
    pred_train=lr.predict(features_train)
    pred_test=lr.predict(features_test)
    print(f"at random state {i}, the training accuracy is:- {r2_score(target_train,pred_train)}")
    print(f"at random state {i}, the testing accuracy is:- {r2_score(target_test,pred_test)}")
```

```
at random state 0, the training accuracy is:- 0.9058706722803965
at random state 0, the testing accuracy is:- 0.8977317673785854
at random state 1, the training accuracy is:- 0.9052277082152647
at random state 1, the testing accuracy is:- 0.9003132716690803
at random state 2, the training accuracy is:- 0.9056858398935238
at random state 2, the testing accuracy is:- 0.8984735750686543
at random state 3, the training accuracy is:- 0.9044956724241888
at random state 3, the testing accuracy is:- 0.9032713403489678
at random state 4, the training accuracy is:- 0.9031354749784841
at random state 4, the testing accuracy is:- 0.9085937883962479
at random state 5, the training accuracy is:- 0.9026338340566922
at random state 5, the testing accuracy is:- 0.9101290251842342
at random state 6, the training accuracy is:- 0.9028087364691222
at random state 6, the testing accuracy is:- 0.9097783155230829
at random state 7, the training accuracy is:- 0.9030731852162728
at random state 7, the testing accuracy is:- 0.9090152662295625
at random state 8, the training accuracy is:- 0.9031695399147395
at random state 8, the testing accuracy is:- 0.9089840252887
at random state 9, the training accuracy is:- 0.9034746951288706
```

From this we can choose the best score from the random state

12 looks good since the ratio between the training and testing score is close

features_train,features_test,target_train,target_test=train_test_split(features,target,test_size=0.2,random_state=12)

- Training the model:
  lr.fit(features_train,target_train)
  pred_test=lr.predict(features_test)
  print(r2_score(target_test,pred_test))

we get the r2 score and we will check the cross validation score to check if the model is overfitting

- Checking the cross validation:
  We can validate the performace of the model
  from sklearn.model_selection import cross_val_score
  Train_accuracy=r2_score(target_train,pred_train)
  Test_accuracy=r2_score(target_test,pred_test)

  ```
  for j in range(2,10):
      cv_score=cross_val_score(lr,features,target,cv=j)
      cv_mean=cv_score.mean()
      print(f"at cross fold {j} the cv score is  {cv_mean} and accuracy score for training is {Train_accuracy} and accuracy for the testing is {Test_accuracy}")
  ```

```
print("\n")
```

taking the cv score as 5 keeping 20:80 ratio, if I take my cv as 5 it means my model will be divided into 5 parts and every part will take random data and my cv method will check the accuracy or the performance for 5 times , and we will take the mean , after that we will within the range 2 to 10 what is the cv score

- Regularization:
  ```
  from sklearn.model_selection import GridSearchCV
  from sklearn.model_selection import cross_val_score
  import warnings
  warnings.filterwarnings('ignore')
  ```

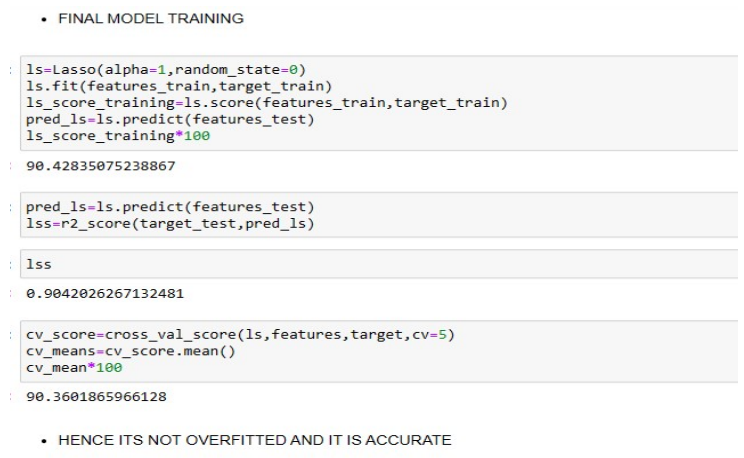
there are two regularization technique: ridge and lasso

when we use lasso there are two parameters alpha and random state

to make the combination between the values of the parameters and it will give you the best parameter that maximize the performance of the model

```
from sklearn.linear_model import Lasso

parameters={'alpha':[.0001,.001,.01,.1,1,10],

        'random_state':list(range(0,10))}

ls=Lasso()

clf=GridSearchCV(ls,parameters)

clf.fit(features_train,target_train)

print(clf.best_params_)
```

- Final model training:
  We will use both parameters to frame my model
  Lss score is the training score



- FINAL MODEL TRAINING

```
ls=Lasso(alpha=1,random_state=0)
ls.fit(features_train,target_train)
ls_score_training=ls.score(features_train,target_train)
pred_ls=ls.predict(features_test)
ls_score_training*100
```

```
90.42835075238867
```

```
pred_ls=ls.predict(features_test)
lss=r2_score(target_test,pred_ls)
```

```
lss
```

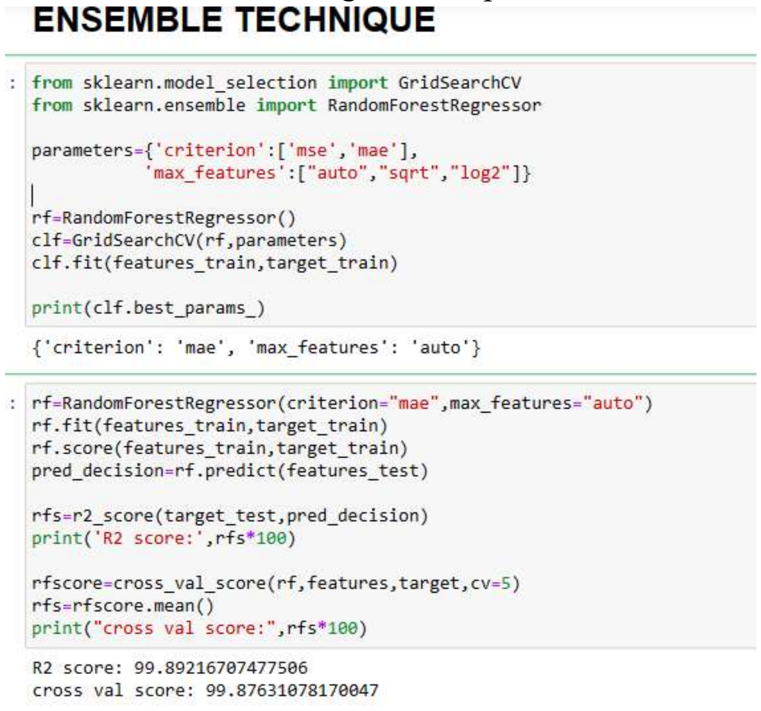
```
0.9042026267132481
```

```
cv_score=cross_val_score(ls,features,target,cv=5)
cv_means=cv_score.mean()
cv_mean*100
```

```
90.3601865966128
```

- HENCE ITS NOT OVERFITTED AND IT IS ACCURATE

- Ensemble techniques:
  Here we are using randomforest and using two parameters mse and mae and we are selecting the best parameters



ENSEMBLE TECHNIQUE

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

parameters={'criterion':['mse','mae'],
           'max_features':["auto","sqrt","log2"]}

rf=RandomForestRegressor()
clf=GridSearchCV(rf,parameters)
clf.fit(features_train,target_train)

print(clf.best_params_)
```

```
{'criterion': 'mae', 'max_features': 'auto'}
```

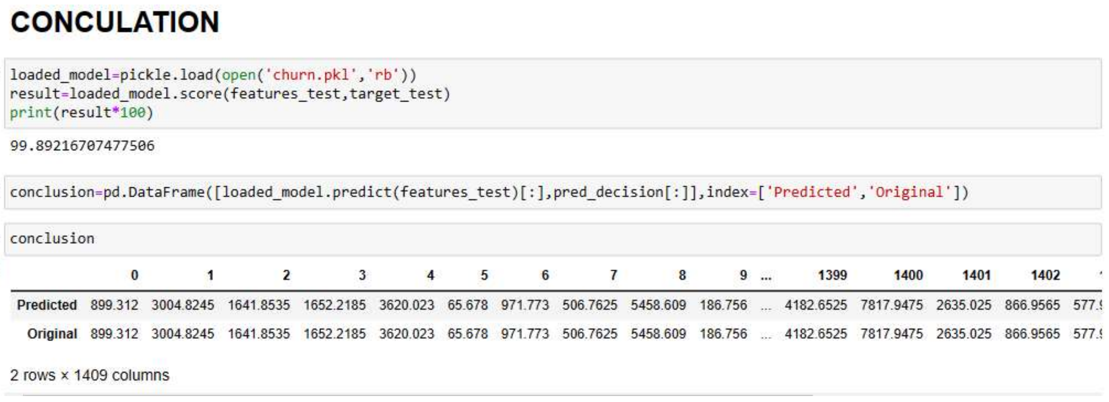
```
rf=RandomForestRegressor(criterion="mae",max_features="auto")
rf.fit(features_train,target_train)
rf.score(features_train,target_train)
pred_decision=rf.predict(features_test)

rfs=r2_score(target_test,pred_decision)
print('R2 score:',rfs*100)

rfscore=cross_val_score(rf,features,target,cv=5)
rfs=rfscore.mean()
print("cross val score:",rfs*100)
```

```
R2 score: 99.89216707477506
cross val score: 99.87631078170047
```

WE ARE GETTING MODEL ACCURACY AND CROSS VALIDATION BOTH AS 99% WHICH SHOWS OUR MODEL IS PERFORMING WELL

- saving the model:
  import pickle
  filename='churn.pkl'
  pickle.dump(rf,open(filename,'wb'))

5.CONCLUSION:



CONCULATION

```
loaded_model=pickle.load(open('churn.pkl','rb'))
result=loaded_model.score(features_test,target_test)
print(result*100)
```

```
99.89216707477506
```

```
conclusion=pd.DataFrame([loaded_model.predict(features_test)[:],pred_decision[:]],index=['Predicted','Original'])
```

```
conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	1399	1400	1401	1402	
Predicted	899.312	3004.8245	1641.8535	1652.2185	3620.023	65.678	971.773	506.7625	5458.609	186.756	...	4182.6525	7817.9475	2635.025	866.9565	577.
Original	899.312	3004.8245	1641.8535	1652.2185	3620.023	65.678	971.773	506.7625	5458.609	186.756	...	4182.6525	7817.9475	2635.025	866.9565	577.

2 rows × 1409 columns

We are getting randomforest as my best model

Here the predicted value is equal to my original value that’s why I am getting an accuracy of 99% so the score is very good