```
In [0]: import pandas as pd
        from sklearn.metrics import accuracy_score
        from collections import Counter
        from sklearn.feature_selection import SelectKBest
        from sklearn.feature_selection import f_classif
        from sklearn.model_selection import train_test_split
        from sklearn.svm import LinearSVC
        from sklearn.preprocessing import StandardScaler
        from sklearn.preprocessing import LabelEncoder
        from sklearn.feature_selection import RFE
        from sklearn.svm import SVC
        from sklearn.model_selection import RandomizedSearchCV
```

```
In [0]: Data=pd.read_csv("Churn_Modelling.csv")
```

Preprocessing

```
In [61]: Data.dtypes

Out[61]: RowNumber            int64
         CustomerId           int64
         Surname             object
         CreditScore          int64
         Geography           object
         Gender              object
         Age                  int64
         Tenure               int64
         Balance            float64
         NumOfProducts        int64
         HasCrCard            int64
         IsActiveMember       int64
         EstimatedSalary    float64
         Exited               int64
         dtype: object
```

```
In [62]: Data.isnull().sum()

Out[62]: RowNumber          0
         CustomerId         0
         Surname            0
         CreditScore        0
         Geography          0
         Gender             0
         Age                0
         Tenure             0
         Balance            0
         NumOfProducts      0
         HasCrCard          0
         IsActiveMember     0
         EstimatedSalary    0
         Exited             0
         dtype: int64
```

```
In [63]: Data.head()
```

Out[63]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCar
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	

```
In [0]: len(Counter(Data.Surname))#Checking the number of unique values in Surname column

Out[0]: 2932
```

```
In [0]: Counter(Data.Geography)#Checking unique values in Geography column

Out[0]: Counter({'France': 5014, 'Germany': 2509, 'Spain': 2477})
```

```
In [0]: len(Counter(Data.CustomerId))#Checking whether CustomerId's are unique

Out[0]: 10000
```

```
In [0]: import numpy as np
        print(np.var(Data.Balance),np.var(Data.EstimatedSalary))
        #Balance and EstimatedSalary has high variance

        3893046832.3731775 3307126038.456105
```

```
In [0]: sc=StandardScaler()
        lb=LabelEncoder()
        Data.Balance=sc.fit_transform(Data.Balance.values.reshape(-1,1))
        Data.EstimatedSalary=sc.fit_transform(Data.EstimatedSalary.values.reshape(-1,1))
```

```
In [75]: Data.Geography=lb.fit_transform(Data.Geography)
         print(list(zip(lb.classes_,lb.transform(lb.classes_))))

         [('France', 0), ('Germany', 1), ('Spain', 2)]
```

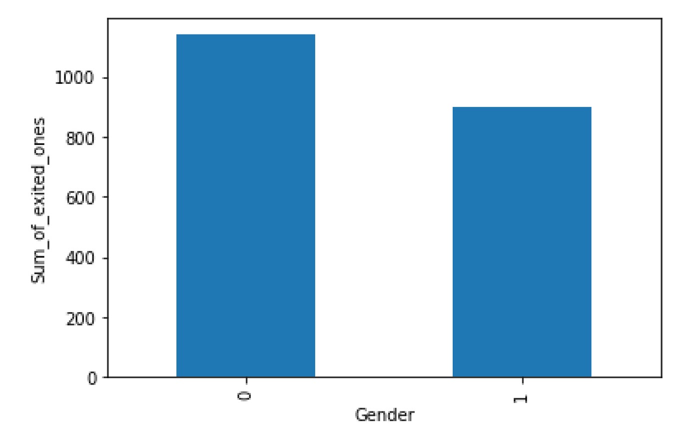
```
In [125]: lb1=LabelEncoder()
          Data.Gender=lb1.fit_transform(Data.Gender)
          print(list(zip(lb1.classes_,lb1.transform(lb1.classes_))))#Female:0,Male:1

          [(0, 0), (1, 1)]
```

```
In [0]: len(Counter(Data.CreditScore))

Out[0]: 460
```

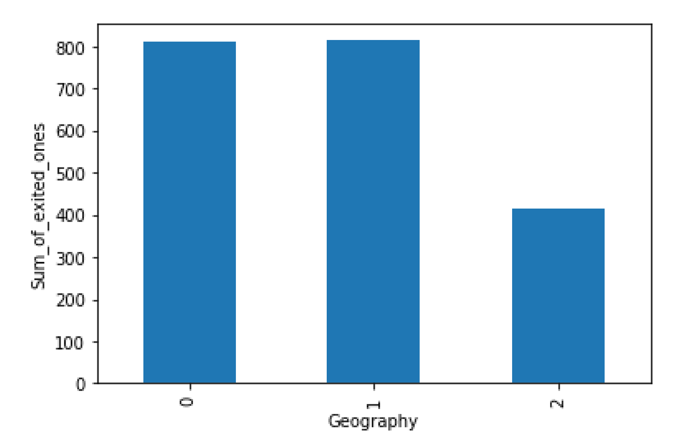
Visualization

```
In [0]: import matplotlib.pyplot as plt
```

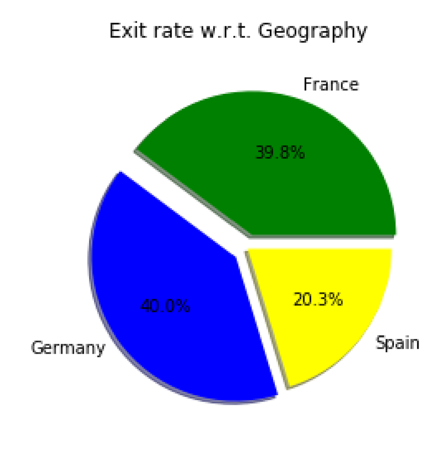
```
In [128]: Data.groupby("Gender")["Exited"].sum().plot("bar")#This shows that majority who exited where males
          plt.ylabel("Sum_of_exited_ones")

Out[128]: Text(0, 0.5, 'Sum_of_exited_ones')
```




```
In [127]: Data.groupby("Geography")["Exited"].sum().plot(kind="bar")#(0=France,1=Germany,2=Spain)
          #Majority those who exited where from germany
          plt.ylabel("Sum_of_exited_ones")

Out[127]: Text(0, 0.5, 'Sum_of_exited_ones')
```




```
In [0]: sizes=list(Data.groupby("Geography")["Exited"].sum()/Data.groupby("Geography")["Exited"].sum().sum
        ())
        labels=['France','Germany','Spain']
        explode = (0.1, 0.1, 0)
        colors=['green','blue','yellow']
        plt.pie(sizes,explode=explode, labels=labels,colors=colors,autopct='%1.1f%%', shadow=True)
        plt.suptitle("Exit rate w.r.t. Geography")

Out[0]: Text(0.5, 0.98, 'Exit rate w.r.t. Geography')
```




```
In [0]: Data_new=Data.drop(["RowNumber","CustomerId","Surname","CreditScore"],1)
```

```
In [82]: Data_new.corr()#Inverse relationship between CreditScore and Exited found
```

Out[82]:

	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estim
Geography	1.000000	0.004719	0.022812	0.003739	0.069408	0.003972	-0.008523	0.006724	
Gender	0.004719	1.000000	-0.027544	0.014733	0.012087	-0.021859	0.005766	0.022544	
Age	0.022812	-0.027544	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	
Tenure	0.003739	0.014733	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	
Balance	0.069408	0.012087	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	
NumOfProducts	0.003972	-0.021859	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	
HasCrCard	-0.008523	0.005766	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	
IsActiveMember	0.006724	0.022544	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	
EstimatedSalary	-0.001369	-0.008112	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	
Exited	0.035943	-0.106512	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	

```
In [0]: Counter(Data_new.Exited)# It is a little balanced dataset

Out[0]: Counter({0: 7963, 1: 2037})
```

```
In [67]: Data_new.shape

Out[67]: (10000, 10)
```

Feature Selection using BestSubset Selection

```
In [0]: X=Data_new.iloc[:,0:10]
        y=Data_new.Exited
```

```
In [0]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)
```

```
In [97]: test = SelectKBest(score_func=f_classif, k=4)
         fit = test.fit(X_train, y_train)

         /usr/local/lib/python3.6/dist-packages/sklearn/feature_selection/univariate_selection.py:115: Ru
         ntimeWarning: divide by zero encountered in true_divide
           f = msb / msw
```

```
In [98]: print(fit.scores_)
         features = fit.transform(X)
         print(features[0:5,:])

         [1.25359982e+01 8.07028414e+01 6.07549620e+02 5.29494469e+00
          1.06086561e+02 2.77449856e+01 5.89288302e-01 1.44507348e+02
          1.43683198e+00            inf]
         [[42.         -1.22584767  1.          1.        ]
          [41.          0.11735002  1.          0.        ]
          [42.          1.33305335  0.          1.        ]
          [39.         -1.22584767  0.          0.        ]
          [43.          0.7857279   1.          0.        ]]
```

```
In [136]: model=SVC(kernel="linear",gamma='scale')
          rfe = RFE(model, 3)
          fit=rfe.fit(X_train,y_train)
          print("Num Features: %s" % (fit.n_features_))
          print("Selected Features: %s" % (fit.support_))
          print("Feature Ranking: %s" % (fit.ranking_))

          Num Features: 3
          Selected Features: [False False False False False False False  True  True  True]
          Feature Ranking: [6 5 8 4 3 2 7 1 1 1]
```

```
In [0]: #Recursive feature elimination gives last three i.e. HasCrCard,IsActiveMember,EstimatedSalary contri
        bute most to the target variable
        X_train_new=X_train[["HasCrCard","IsActiveMember","EstimatedSalary"]]
        X_test_new=X_test[["HasCrCard","IsActiveMember","EstimatedSalary"]]
```

```
In [124]: model.fit(X_train_new,y_train)
          pred=model.predict(X_test_new)
          acc=accuracy_score(y_test,pred)
          print(acc)

          0.793
```

```
In [123]: model1=LinearSVC(penalty='l1',loss='l2',dual=False)
          model1.fit(X_train_new,y_train)
          pred1=model1.predict(X_test_new)
          acc1=accuracy_score(y_test,pred1)
          print(acc1)
          #We get same accuracy in case of regularization as well as Recursive Feature Elimination

          0.793

          /usr/local/lib/python3.6/dist-packages/sklearn/svm/classes.py:220: DeprecationWarning: loss='l2'
          has been deprecated in favor of loss='squared_hinge' as of 0.16. Backward compatibility for the
          loss='l2' will be removed in 1.0
            DeprecationWarning)
```

RandomisedSearch

```
In [137]: param_dict={'coef0':[0.0,2.0,3.0],'kernel':["linear","rbf"],'degree':[3,5,1]}
          rs=RandomizedSearchCV(model,param_distributions=param_dict, n_iter=10)
          rs.fit(X_train_new,y_train)

          /usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:1978: FutureWarning: Th
          e default value of cv will change from 3 to 5 in version 0.22. Specify it explicitly to silence
          this warning.
            warnings.warn(CV_WARNING, FutureWarning)

Out[137]: RandomizedSearchCV(cv='warn', error_score='raise-deprecating',
                             estimator=SVC(C=1.0, cache_size=200, class_weight=None,
                                           coef0=0.0, decision_function_shape='ovr',
                                           degree=3, gamma='scale', kernel='linear',
                                           max_iter=-1, probability=False,
                                           random_state=None, shrinking=True, tol=0.001,
                                           verbose=False),
                             iid='warn', n_iter=10, n_jobs=None,
                             param_distributions={'coef0': [0.0, 2.0, 3.0],
                                                  'degree': [3, 5, 1],
                                                  'kernel': ['linear', 'rbf']},
                             pre_dispatch='2*n_jobs', random_state=None, refit=True,
                             return_train_score=False, scoring=None, verbose=0)
```

```
In [138]: rs.best_params_#This was the model that we use in SVC and got an accuracy of 0.793 or 79.3%
```