Feature Selection Based On Univariant (ANOVA) test for classification.

What is the Univariant test ?

- The elimination process aims to reduce the size of the input feature set and at the same time to retain the class discreminatory information for classification problems.In simple word unwanted feature are removed for getting the proper and input features and at the same time it we get the proper information for classification problem. In that we taking the analysis of the variance i.e.ANOVA is can be thought as an extension of the T-test Thde independent T-test is used to comapare the mean of the condition between two groups although this ANOVA test is based on F-Test.F-Test is any stastical test in which test stastic has an F-distributions under the null hypothesis.

- An F-Test is any stastical test in which the test stastic has an F-distribution under the null hypothesis.

- Analysis of variance (ANOVA) is collection of stastical models and their associated estimation procedures(such as the 'variation' among and between groups) used to analyze the diffrences among group means in a sample

- In the T-test we compare the mean from the two groups but in ANOVA we compre the mean of the groups which two or more than two groups

- F-test is used to comparing the factors of the total deviation.For example,in one-way or single factor ANOVA stastical significance is tested for by comparing the F-test stastic.

- The ANOVA was developed by stastacian Rounald Fisher that is also known as F-test . The ANOVA is based on the law of the total variance where the observed variance in particular variable particant into the attribute to the diffrent source of the variation.
- F = variance between the features/variance within the features

- We having the choices of ANOVA according to that we have to use it like f_classif,f_regression.

Classification:-

In [36]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

In [37]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import f_classif,f_regression
from sklearn.feature_selection import SelectKBest,SelectPercentile
```

In [ ]:

In [38]:
```
data = pd.read_csv('santander-train.csv',nrows=20000)
data.head()
```

Out[38]:

	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_op_var40_comer_ult1	imp_op_var40_comer_ult3
0	1	2	23	0.0	0.0	0.0	0.0	0.0
1	3	2	34	0.0	0.0	0.0	0.0	0.0
2	4	2	23	0.0	0.0	0.0	0.0	0.0
3	8	2	37	0.0	195.0	195.0	0.0	0.0
4	10	2	39	0.0	0.0	0.0	0.0	0.0

5 rows × 371 columns

In [39]:
```
x = data.drop('TARGET',axis =1)
y = data['TARGET']
x.shape , y.shape
```

Out[39]:
```
((20000, 370), (20000,))
```

In [40]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size =.8,random_state =0,stratify=y)
```

Remove the constant,Quasi constant and correlated features

In [41]:
```
#remove the constant and quasi constant
constant_filter = VarianceThreshold(0.01)
constant_filter.fit(x_train)
x_train_filter = constant_filter.transform(x_train)
x_test_filter = constant_filter.transform(x_test)
```

In [42]:
```
x_train_filter.shape,x_test_filter.shape
```

Out[42]:
```
((16000, 245), (4000, 245))
```

In [ ]:
```
#Remove the duplicate features
```

In [43]:
```
x_train_T= x_train_filter.T
x_test_T = x_test_filter.T
```

In [44]:
```
x_train_T = pd.DataFrame(x_train_T)
x_test_T  = pd.DataFrame(x_test_T)
```

In [45]:
```
x_train_T.duplicated().sum()
```

Out[45]:
```
18
```

In [46]:
```
duplicated_features = x_train_T.duplicated()
```

In [47]:
```
features_to_keep = [not index for index in duplicated_features ]
```

In [48]:
```
features_to_keep = [not index for index in duplicated_features ]
x_train_unique = x_train_T[features_to_keep].T
x_test_unique = x_test_T[features_to_keep].T
```

In [49]:
```
x_train_unique.shape,x_train.shape
```

Out[49]:
```
((16000, 227), (16000, 370))
```

In [ ]:

NOW DO F-Test

In [50]:
```
sel = f_classif(x_train_unique,y_train)
sel
#we got the two arrays that are the f and p array
```

Out[50]:
```
(array([3.42911520e-01, 1.22929093e+00, 1.61291330e+02, 4.01025132e-01,
        8.37661151e-01, 2.39279390e-03, 4.41633351e-02, 1.36337510e-01,
        1.84647123e+00, 2.03640367e+00, 7.98057954e-03, 1.14063993e+00,
        6.32266614e-03, 1.55626237e+01, 1.53553790e+01, 1.28615978e+01,
        1.61834746e+01, 1.59638013e+01, 1.21977511e+01, 9.03776687e-02,
        1.00443179e+00, 1.53946148e+01, 2.50428951e+02, 2.98696944e+01,
        1.06266841e+01, 2.63630437e+01, 1.66417611e+01, 3.13699473e+01,
        2.47256550e+01, 2.60021376e+01, 3.26742018e+01, 9.94259060e+00,
        1.48208220e+01, 1.50040146e+01, 1.34739830e+01, 7.03118653e+00,
        1.36234772e+01, 7.95962134e+00, 3.15161070e+02, 1.79631284e+00,
        1.66910747e+00, 1.21138302e+01, 1.10928892e+01, 1.00443179e+00,
        2.31851572e+00, 8.93973153e+01, 7.53868668e+00, 2.38490562e+02,
        2.98696944e+01, 1.06266841e+01, 2.61694409e+01, 1.66053267e+01,
        2.93013259e+01, 2.44433356e+01, 2.60021376e+01, 5.59623841e+00,
        5.65080530e+00, 3.11715028e+01, 9.94259060e+00, 6.69237272e-01,
        6.73931889e-01, 5.91355150e-01, 2.16653744e+00, 1.57036464e+00,
        1.48180592e+01, 1.50040146e+01, 4.10147572e+00, 5.08119829e+00,
        2.86061739e-01, 4.74076524e-04, 3.22895933e-02, 3.61497992e+00,
        2.62641383e-01, 1.44465136e+00, 2.39577575e+00, 3.25151692e+00,
        2.66120176e-01, 1.33584657e+00, 2.15986976e+00, 2.95680783e+01,
        2.74320562e+02, 1.79136749e+00, 1.65942415e+00, 4.55732338e-01,
        8.03423196e+01, 5.33753163e+00, 3.43569515e+00, 5.38991827e+00,
        6.48705021e+00, 1.14907051e+01, 2.46676043e+02, 1.48964854e+00,
        1.48528608e+00, 1.35499717e+00, 5.04105291e+00, 8.00857735e-02,
        5.92081628e-01, 7.49538059e+00, 1.43768803e+01, 3.96797511e+00,
        1.84630418e+01, 5.93034025e-01, 6.23117305e-02, 1.32846978e-01,
        7.36058444e+00, 4.67453255e-01, 6.53434886e-01, 2.32603599e+01,
        8.82160365e-02, 4.03681937e-01, 1.12281656e-01, 1.22229167e+00,
        9.50849020e+00, 3.31504999e-01, 1.52799424e+02, 9.58201843e-01,
        3.81283407e-01, 8.05456673e-01, 2.11768899e-01, 4.23427422e-02,
        4.23427422e-02, 4.23427422e-02, 4.23675848e-01, 9.58201843e-01,
        8.05456673e-01, 4.23675848e-01, 7.83475034e+00, 7.84514734e-01,
        4.28901812e-02, 1.44260945e-01, 4.33508271e-02, 4.23427422e-02,
        3.34880062e-02, 1.90957786e-01, 4.06328805e-01, 1.70136127e-01,
        4.23427422e-02, 5.36587189e-01, 1.87563339e+00, 4.23427422e-02,
        4.23427422e-02, 4.23427422e-02, 1.25864897e-01, 1.50227029e-01,
        7.58252261e-01, 3.69870284e-01, 6.31366809e-02, 1.39484806e+00,
        5.24649450e+00, 8.74444426e-02, 1.20564528e+01, 1.08123286e+00,
        8.46910021e-02, 2.36606015e-01, 5.89389684e+00, 2.77252663e-01,
        4.15074036e-01, 1.44558159e-01, 1.17723957e+00, 9.22407334e-01,
        1.45895164e+01, 1.86656969e+00, 5.43234215e+00, 1.86971763e-02,
        3.09123385e+02, 7.12088878e+00, 1.49660894e+01, 2.43275497e+01,
        4.52466899e+00, 2.03980835e-01, 5.87673213e-03, 4.98543138e-02,
        5.16359722e-02, 1.09646850e-01, 2.06155459e+00, 2.99184059e+00,
        2.21995621e-02, 1.13858713e-01, 1.14255501e+01, 1.13785982e+01,
        1.19082872e+01, 1.18528440e+01, 2.65465286e-02, 1.52894509e-01,
        4.63685902e+00, 2.10080736e+00, 1.65523608e-01, 2.16891078e-01,
        1.40302586e+00, 5.48359285e-01, 6.35218588e-02, 4.88987865e+00,
        2.49656443e+00, 4.58216058e+00, 4.15099427e+00, 4.56305342e-01,
        1.66491238e-01, 3.90777488e-01, 3.50953637e-01, 5.52484208e+00,
        2.37194124e+00, 7.35792170e+00, 7.47930913e+00, 1.19139338e+01,
        3.63667170e+00, 1.46817492e+01, 1.40921857e+01, 2.55113543e+00,
        7.93363123e-01, 2.95584767e+00, 2.83339311e+00, 4.73780486e-02,
        4.26696894e-02, 6.24420202e-02, 6.13788649e-02, 5.70774760e-02,
        7.65160310e-02, 1.10327676e-01, 1.26598304e-01, 4.23427422e-02,
        1.11726086e-01, 1.17106404e-01, 3.13117156e-01, 1.24267517e-01,
        2.84184735e-01, 3.29540269e-01, 1.12297080e+01]),
 array([5.58161700e-01, 2.67561647e-01, 8.89333290e-37, 5.26569363e-01,
        3.60080335e-01, 9.60986695e-01, 8.33552698e-01, 7.11954403e-01,
        1.74213527e-01, 1.53591870e-01, 9.28817521e-01, 2.85533263e-01,
        9.36623841e-01, 8.01575252e-05, 8.94375507e-05, 3.36393721e-04,
        5.77577141e-05, 6.48544590e-05, 4.79763179e-04, 7.63701483e-01,
        3.16255673e-01, 8.76012543e-05, 5.56578484e-56, 4.68990120e-08,
        1.11700314e-03, 2.86219940e-07, 4.53647534e-05, 2.16766394e-08,
        6.67830586e-07, 3.44933857e-07, 1.10916535e-08, 1.61796584e-03,
        1.18682969e-04, 1.07709938e-04, 2.42680916e-04, 8.01812206e-03,
        2.24116226e-04, 4.78913410e-03, 7.66573763e-70, 1.80177928e-01,
        1.96396787e-01, 5.01825968e-04, 8.68554202e-04, 3.16255673e-01,
        1.27861727e-01, 3.66783202e-21, 6.04554908e-03, 2.03825983e-53,
        4.68990120e-08, 1.11700314e-03, 3.16348432e-07, 4.62436764e-05,
        6.28457802e-08, 7.73029885e-07, 3.44933857e-07, 1.80109375e-02,
        1.74590458e-02, 2.40048097e-08, 1.61796584e-03, 4.13329839e-01,
        4.11696353e-01, 4.41906921e-01, 1.41063166e-01, 2.10172382e-01,
        1.18856798e-04, 1.07709938e-04, 4.28623726e-02, 2.42001211e-02,
        5.92762818e-01, 9.82629065e-01, 8.57395823e-01, 5.72793629e-02,
        6.08318344e-01, 2.29405921e-01, 1.21683164e-01, 7.13761984e-02,
        6.05953475e-01, 2.47785024e-01, 1.41676361e-01, 5.47783585e-08,
        4.18717532e-61, 1.80778657e-01, 1.97699723e-01, 4.99635011e-01,
        3.49020462e-19, 2.08836652e-02, 6.38201266e-02, 2.02659144e-02,
        1.08755946e-02, 7.01140336e-04, 3.55791184e-55, 2.22288988e-01,
        2.22967265e-01, 2.44423761e-01, 2.47670413e-02, 7.77184868e-01,
        4.41626654e-01, 6.19258331e-03, 1.50177954e-04, 4.63904749e-02,
        1.74254389e-05, 4.41259644e-01, 8.02881978e-01, 7.15503084e-01,
        6.67404495e-03, 4.94171032e-01, 4.18899308e-01, 1.42788438e-06,
        7.66461321e-01, 5.25202939e-01, 7.37565700e-01, 2.68928005e-01,
        2.04871607e-03, 5.64782330e-01, 6.11812415e-35, 3.27655140e-01,
        5.36925966e-01, 3.69480411e-01, 6.45390733e-01, 8.36970444e-01,
        8.36970444e-01, 8.36970444e-01, 5.15117892e-01, 3.27655140e-01,
        3.69480411e-01, 5.15117892e-01, 5.13125866e-03, 3.75777233e-01,
        8.35934829e-01, 7.04086312e-01, 8.35068733e-01, 8.36970444e-01,
        8.54802468e-01, 6.62126552e-01, 5.23847887e-01, 6.79996382e-01,
        8.36970444e-01, 4.63861289e-01, 1.70850496e-01, 8.36970444e-01,
        8.36970444e-01, 8.36970444e-01, 7.22763262e-01, 6.98323652e-01,
        3.83889093e-01, 5.43083617e-01, 8.01608490e-01, 2.37605638e-01,
        2.20039777e-02, 7.67455348e-01, 5.17497373e-04, 2.98437634e-01,
        7.71041949e-01, 6.26674911e-01, 1.52044006e-02, 5.98514898e-01,
        5.19414532e-01, 7.03796034e-01, 2.77935032e-01, 3.36858193e-01,
        1.34160887e-04, 1.71887678e-01, 1.97795129e-02, 8.91239914e-01,
        1.49493801e-68, 7.62677506e-03, 1.09894509e-04, 8.20840983e-07,
        3.34247932e-02, 6.51532760e-01, 9.38895107e-01, 8.23319823e-01,
        8.20243529e-01, 7.40550887e-01, 1.51075546e-01, 8.37042875e-02,
        8.81559318e-01, 7.35797526e-01, 7.26140977e-04, 7.44713683e-04,
        5.60290090e-04, 5.77206713e-04, 8.70574777e-01, 6.95789674e-01,
        3.13071224e-02, 1.47240985e-01, 6.84126595e-01, 6.41425380e-01,
        2.36235196e-01, 4.58999733e-01, 8.01016915e-01, 2.70286705e-02,
        1.14114721e-01, 3.23215275e-02, 4.16265059e-02, 4.99365521e-01,
        6.83254631e-01, 5.31899920e-01, 5.53582168e-01, 1.87603701e-02,
        1.23553129e-01, 6.68393143e-03, 6.24807382e-03, 5.58595551e-04,
        5.65377099e-02, 1.27759206e-04, 1.74681308e-04, 1.10234773e-01,
        3.73098519e-01, 8.55867690e-02, 9.23426211e-02, 8.27692918e-01,
        8.36351110e-01, 8.02680255e-01, 8.04332893e-01, 8.11179263e-01,
        7.82079079e-01, 7.39775741e-01, 7.21990132e-01, 8.36970444e-01,
        7.38191886e-01, 7.32198763e-01, 5.75781477e-01, 7.24455979e-01,
        5.93978825e-01, 5.65937990e-01, 8.06846135e-04]))
```

In [ ]:
```
#the p values which less than the 0.05 then this kind p values are very important.
```

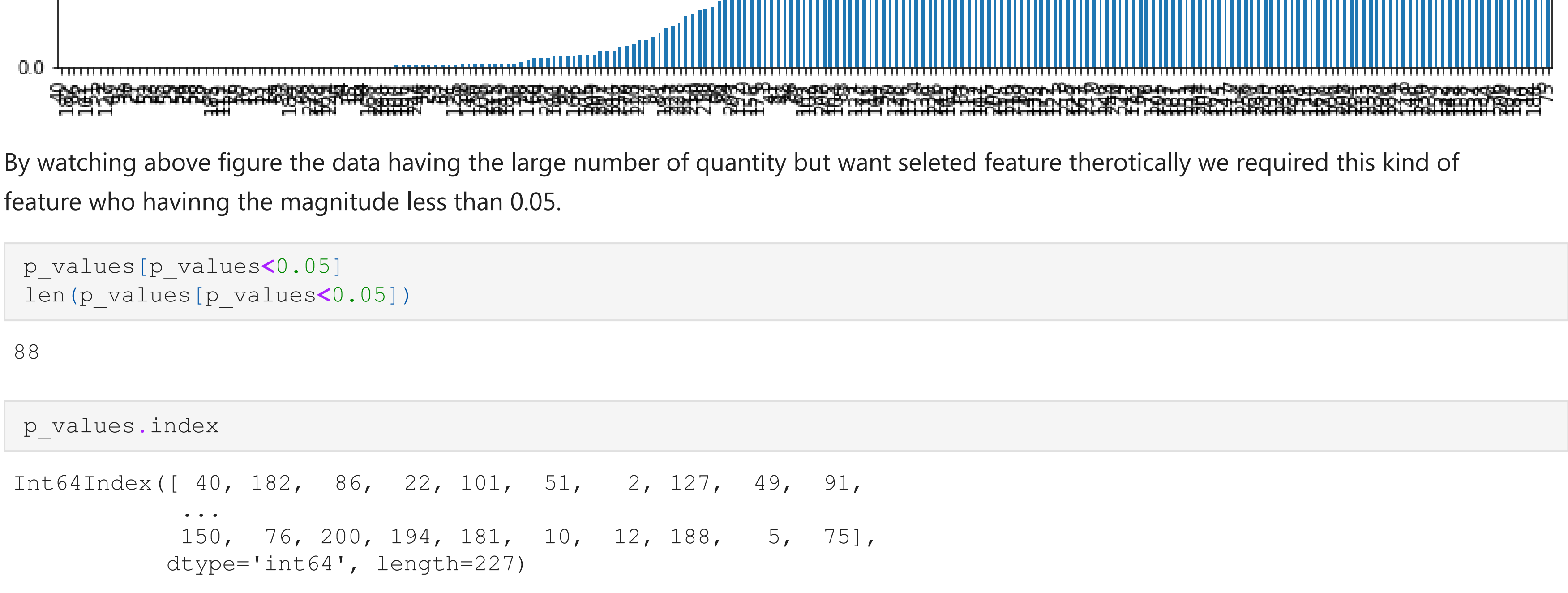
In [51]:
```
import pandas as pd
p_values = pd.Series(sel[1])
p_values.index = x_train_unique.columns
p_values.sort_values(ascending=True,inplace=True)
```

In [52]:
```
len(p_values)
```

Out[52]:
```
227
```

In [53]:
```
p_values.plot.bar(figsize = (15,5))
```

Out[53]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xeb32a90>
```



By watching above figure the data having the large number of quantity but want seleted feature therotically we required this kind of feature who havinng the magnitude less than 0.05.

In [27]:
```
p_values[p_values<0.05]
len(p_values[p_values<0.05])
```

Out[27]:
```
88
```

In [54]:
```
p_values.index
```

Out[54]:
```
Int64Index([ 40, 182,  86,  22, 101,  51,   2, 127,  49,  91,
            ...
            150,  76, 200, 194, 181,  10,  12, 188,   5,  75],
           dtype='int64', length=227)
```

In [55]:
```
x_train_p = x_train_unique[p_values.index]
x_test_p  = x_test_unique[p_values.index]
```

In [56]:
```
def randomforest(x_train,x_test,y_train,y_test):
    clf = RandomForestClassifier(random_state=0,n_jobs=-1,n_estimators=1000)
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test)
    print('Accuracy :' , accuracy_score(y_test,y_pred))
```

In [57]:
```
%%time
randomforest(x_train_p,x_test_p,y_train,y_test)
```

```
Accuracy : 0.9585
Wall time: 23.4 s
```

In [62]:
```
%%time
randomforest(x_train,x_test,y_train,y_test)
```

```
Accuracy : 0.959
Wall time: 29 s
```

If we observed that there is time that magic time taken for the computation is higher than the previous if we use the whole data in train and test split.

In [ ]: