Thor Project (1)

February 26, 2018

In [1]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
import scipy.stats as stats
import itertools
```

```
//anaconda/lib/python3.5/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: The pand
  from pandas.core import datetools
```

In [2]:
```
data = pd.read_csv("mjollnir.csv")
data.shape
```

Out[2]:
```
(281, 22)
```

In [3]:
```
def getFeaturesCombo(features):
    print("Generating list_of_features")
    list_of_features = []
    for L in range(0, len(features)+1):
        for subset in itertools.combinations(features, L):
            list_of_features.append([s for s in subset])
    print("Got list_of_features")
    return list_of_features

def getResidualsPvalue(X, y, list_of_features, percentile):
    """returns the p-value at a percentile"""
    "Getting all p-values"
    p_values = []
    for i in list_of_features:
        if (len(i) > 0):
            X_subset = sm.add_constant(X[i])
            model = sm.OLS(y, X_subset).fit()
            residuals = model.resid
            p_value = stats.normaltest(residuals)[1]
            p_values.append(p_value)
    pvalue = np.percentile(p_values, percentile)
    print("Got threshold p-value fore residuals")
```

```
            return pvalue

        def getBestModels(X, y, list_of_features,
                          coef_pvalue_threshold,
                          res_pvalue_threshold):
            print("Getting best models")
            winningModels = {}
            #getting all possible combinations of features
            for i in list_of_features:
                if (len(i) > 0):
                    #adding constant to X
                    X_subset = sm.add_constant(X[i])
                    #fitting the model
                    model = sm.OLS(y, X_subset).fit()
                    # 1st condition
                    # Check if p-value of all coefficients is less than 0.05
                    coeffpvalues=model.pvalues
                    coeffpvalues=coeffpvalues.drop(coeffpvalues.index[[0]])
                    if (sum(coeffpvalues < coef_pvalue_threshold) == len(coeffpvalues)):
                        residuals = model.resid
                        #checking if the residuals are normal, a p-value < 0.05
                        #means that the null hypothesis of normality of
                        #residuals is rejected
                        p_value = stats.normaltest(residuals)[1]
                        if (p_value > res_pvalue_threshold):
                            # the number of features is 1, there is no vif score
                            if (len(i) < 2):
                                #storing the feature
                                winningFeatures = [s for s in X_subset.columns]
                                #storing Ajd-R2 as key, list of features as values
                                winningModels[round(model.rsquared_adj, 3)] = winningFeatures
                            else:
                                #getting vifs if list of features is more than 1
                                vifs = [variance_inflation_factor(X_subset.values, i)
                                        for i in range(X_subset.shape[1])]
                                #getting vifs greater than threshold 5
                                above_5_vif = [s for s in vifs if s > 5]
                                # 2nd condition, if no vifs is greater than 5
                                if (len(above_5_vif)) == 0:
                                    print("Model passed, getting features")
                                    #storing the feature
                                    winningFeatures = [s for s in X_subset.columns]
                                    #storing Ajd-R2 as key, list of features as values
                                    winningModels[round(model.rsquared_adj,3)] = winningFeature
            return winningModels

In [4]: features  = ['Max of  likes', 'Max of Categ 2','Max of Categ 10',
                    'Max of Categ 15', 'Max of Categ 17',
```

```
            'Max of Categ 19', 'Max of Categ 20', 'Max of Categ 22',
            'Max of Categ 23', 'Max of Categ 26','Max of Categ 24',
            'Max of # tags', 'Max of Views after 1 day'
            ]

        X = data[features]
        y = data['Max of Views after 5 days']
        coef_pvalue_threshold = 0.05
        list_of_features = getFeaturesCombo(features)

        percentile = 50
        res_pvalue_threshold = getResidualsPvalue(X, y,
                                                  list_of_features,
                                                  percentile)
        winningModels = getBestModels(X, y,
                                      list_of_features,
                                      coef_pvalue_threshold,
                                      res_pvalue_threshold)
```

```
Generating list_of_features
Got list_of_features
Got threshold p-value fore residuals
Getting best models
Model passed, getting features
Model passed, getting features
Model passed, getting features
Model passed, getting features
```

```
In [5]: winningModels
```

```
Out[5]: {0.745: ['const', 'Max of Views after 1 day'],
         0.749: ['const', 'Max of # tags', 'Max of Views after 1 day'],
         0.751: ['const', 'Max of Categ 10', 'Max of Views after 1 day'],
         0.752: ['const',
          'Max of Categ 26',
          'Max of # tags',
          'Max of Views after 1 day'],
         0.757: ['const',
          'Max of Categ 10',
          'Max of # tags',
          'Max of Views after 1 day']}
```