Feature Selection

We all may have faced the problem of identifying the important features from a set of given data and removing the irrelevant or less important features which do not contribute much to our decision making in order to achieve better accuracy for our model.

In machine learning and statistics, feature selection, also known as **variable selection**, **attribute selection** or **variable subset selection**, is the process of reducing the number of input variables when developing a predictive model. Feature selection techniques are used for several reasons:

* It reduces model complexity by dropping some irrelevant features.
* Helps ML algorithm to train a model faster.
* Redcution of dimensionality helps in avoid overfitting.

In this notebook i will be discussing 3 common techniques used for feature selection which are easy to implement and will give you a good results based on problem. Following are the feature selection techniques:

1. **Univariate Selection**
2. **Feature Importance**
3. **Correlation Matrix with Heatmap**

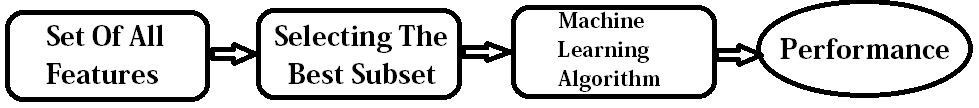
Before Discussing above three techniques let us go through the basic methodologies used for feature selection.

1. Filter Method:

Filter feature selection methods use statistical techniques to evaluate the relationship **between each input variable and the target variable**, and these scores are used as the basis to choose (filter) those input variables that will be used in the model.

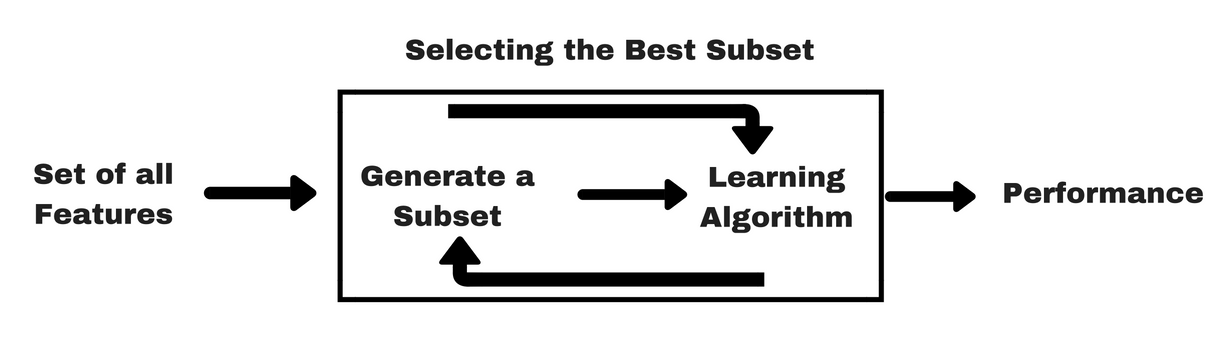
The statistical measures used in filter-based feature selection are generally calculated one input variable at a time with the target variable. As such, they are referred to as **univariate statistical measures**. This may mean that any interaction between input variables is not considered in the filtering process.

**Note:-** In this case, the existence of correlated predictors makes it possible to select important, but redundant, predictors. The obvious consequences of this issue are that too many predictors are chosen and, as a result, collinearity problems arise.



2. Wrapper Method:

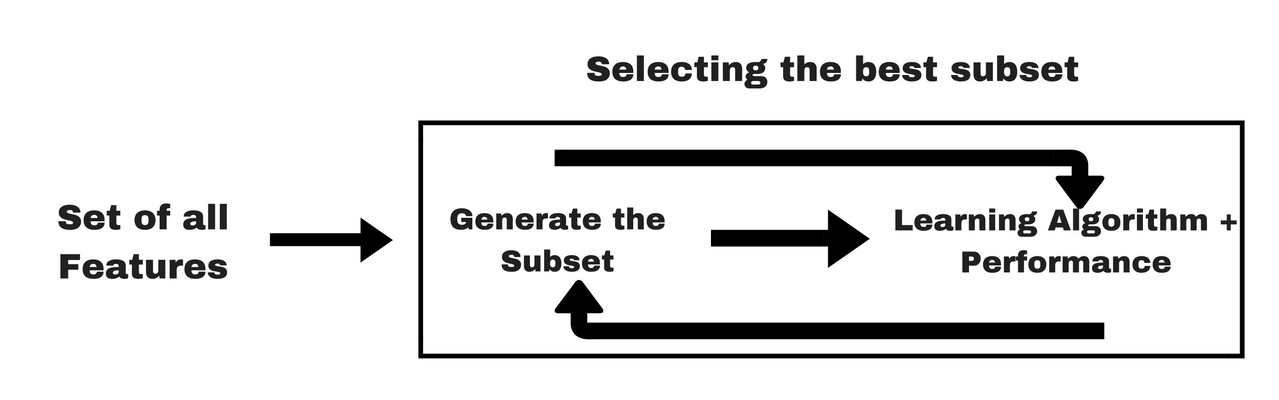
Wrapper feature selection methods create many models with different subsets of input features and select those features that result in the best performing model according to a performance metric. These methods are unconcerned with the variable types, although they can be computationally expensive. Recursive Feature Elimination (RFE) is a good example of a wrapper feature selection method.



1. **Forward Selection**: Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
2. **Backward Elimination**: In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.
3. **Recursive Feature elimination**: It is a greedy optimization algorithm which aims to find the best performing feature subset. It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination.

3. Embedded Method:

Embedded methods combine the qualities’ of filter and wrapper methods.



SUMMARY

We can summarize feature selection as follows.

* **Feature Selection**: Select a subset of input features from the dataset.
  + **Unsupervised**: Do not use the target variable for selecting the feature importance of Input variable (e.g. remove redundant variables).
    - Correlation
  + **Supervised**: Use the target variable (e.g. remove irrelevant I/P features).
    - **Wrapper Method**: Search for well-performing subsets of features.
      * Recursive Feature Elimination (RFE)
    - **Filter Method**: Select subsets of features based on their relationship with the target.
      * Statistical Methods
      * Feature Importance Methods
    - **Intrinsic**: Algorithms that perform automatic feature selection during training.
      * Decision Trees
* **Dimensionality Reduction**: Project input data into a lower-dimensional feature space.

Univariate Method

* Statistical tests can be used to select those features that have the strongest relationship with the output variable.
* The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.
* The example below uses the chi-squared (chi²) statistical test for non-negative features to select k (k=10) of the best features from the Mobile Price Range Prediction Dataset.

In [2]:

import pandas as pd

import numpy as np

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

mobile\_data = pd.read\_csv("../input/mobile-price-classification/train.csv")

X = mobile\_data.iloc[:,0:20] *#independent variables*

y = mobile\_data.iloc[:,-1] *#target variable i.e price range*

In [3]:

mobile\_data.head()

Out[3]:

|  | battery\_power | blue | clock\_speed | dual\_sim | fc | four\_g | int\_memory | m\_dep | mobile\_wt | n\_cores | ... | px\_height | px\_width | ram | sc\_h | sc\_w | talk\_time | three\_g | touch\_screen | wifi | price\_range |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 842 | 0 | 2.2 | 0 | 1 | 0 | 7 | 0.6 | 188 | 2 | ... | 20 | 756 | 2549 | 9 | 7 | 19 | 0 | 0 | 1 | 1 |
| 1 | 1021 | 1 | 0.5 | 1 | 0 | 1 | 53 | 0.7 | 136 | 3 | ... | 905 | 1988 | 2631 | 17 | 3 | 7 | 1 | 1 | 0 | 2 |
| 2 | 563 | 1 | 0.5 | 1 | 2 | 1 | 41 | 0.9 | 145 | 5 | ... | 1263 | 1716 | 2603 | 11 | 2 | 9 | 1 | 1 | 0 | 2 |
| 3 | 615 | 1 | 2.5 | 0 | 0 | 0 | 10 | 0.8 | 131 | 6 | ... | 1216 | 1786 | 2769 | 16 | 8 | 11 | 1 | 0 | 0 | 2 |
| 4 | 1821 | 1 | 1.2 | 0 | 13 | 1 | 44 | 0.6 | 141 | 2 | ... | 1208 | 1212 | 1411 | 8 | 2 | 15 | 1 | 1 | 0 | 1 |

5 rows × 21 columns

In [4]:

*#apply SelectKBest class to extract top 10 best features*

BestFeatures = SelectKBest(score\_func=chi2, k=10)

fit = BestFeatures.fit(X,y)

In [5]:

df\_scores = pd.DataFrame(fit.scores\_)

df\_columns = pd.DataFrame(X.columns)

In [6]:

*#concatenating two dataframes for better visualization*

f\_Scores = pd.concat([df\_columns,df\_scores],axis=1) *# feature scores*

f\_Scores.columns = ['Specs','Score']

In [7]:

f\_Scores *# Score value is directly proportional to the feature importance*

Out[7]:

|  | Specs | Score |
| --- | --- | --- |
| 0 | battery\_power | 14129.866576 |
| 1 | blue | 0.723232 |
| 2 | clock\_speed | 0.648366 |
| 3 | dual\_sim | 0.631011 |
| 4 | fc | 10.135166 |
| 5 | four\_g | 1.521572 |
| 6 | int\_memory | 89.839124 |
| 7 | m\_dep | 0.745820 |
| 8 | mobile\_wt | 95.972863 |
| 9 | n\_cores | 9.097556 |
| 10 | pc | 9.186054 |
| 11 | px\_height | 17363.569536 |
| 12 | px\_width | 9810.586750 |
| 13 | ram | 931267.519053 |
| 14 | sc\_h | 9.614878 |
| 15 | sc\_w | 16.480319 |
| 16 | talk\_time | 13.236400 |
| 17 | three\_g | 0.327643 |
| 18 | touch\_screen | 1.928429 |
| 19 | wifi | 0.422091 |

In [8]:

print(f\_Scores.nlargest(10,'Score')) *# print 10 best features in descending order*

Specs Score

13 ram 931267.519053

11 px\_height 17363.569536

0 battery\_power 14129.866576

12 px\_width 9810.586750

8 mobile\_wt 95.972863

6 int\_memory 89.839124

15 sc\_w 16.480319

16 talk\_time 13.236400

4 fc 10.135166

14 sc\_h 9.614878

Note: Using Above score we can conclude that 'ram' is the most important feature among all the features which is also true in practical scenario.

Feature Importance

* You can get the feature importance of each feature of your dataset by using the feature importance property of the model.
* Feature importance gives you a score for each feature of your data, the higher the score more important or relevant is the feature towards your output variable.
* Feature importance is an inbuilt class that comes with Tree Based Classifiers, but here in this example I will be using XGB Classifier for extracting the top 10 features for the dataset.

In [9]:

import xgboost

import matplotlib.pyplot as plt

model = xgboost.XGBClassifier()

model.fit(X,y)

Out[9]:

XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,

colsample\_bynode=1, colsample\_bytree=1, gamma=0, gpu\_id=-1,

importance\_type='gain', interaction\_constraints='',

learning\_rate=0.300000012, max\_delta\_step=0, max\_depth=6,

min\_child\_weight=1, missing=nan, monotone\_constraints='()',

n\_estimators=100, n\_jobs=0, num\_parallel\_tree=1,

objective='multi:softprob', random\_state=0, reg\_alpha=0,

reg\_lambda=1, scale\_pos\_weight=None, subsample=1,

tree\_method='exact', validate\_parameters=1, verbosity=None)

In [10]:

print(model.feature\_importances\_)

[0.14040743 0.0039343 0.01415134 0.01935592 0.01364646 0.00681664

0.0171691 0.02244249 0.02557429 0.0226691 0.02293218 0.09682195

0.08816101 0.4429328 0.01288087 0.01596808 0.01367627 0.00777803

0.00664631 0.00603543]

In [11]:

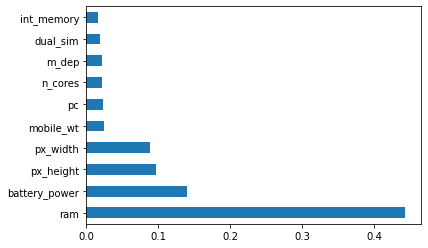
*# plot the graph of feature importances for better visualization*

feat\_imp = pd.Series(model.feature\_importances\_, index=X.columns)

feat\_imp.nlargest(10).plot(kind='barh')

plt.figure(figsize=(8,6))

plt.show()



<Figure size 576x432 with 0 Axes>

Correlation Matrix

* Correlation states how the features are related to each other or the target variable.
* Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable)
* Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.

In [12]:

import seaborn as sns

*#get correlations of each features in dataset*

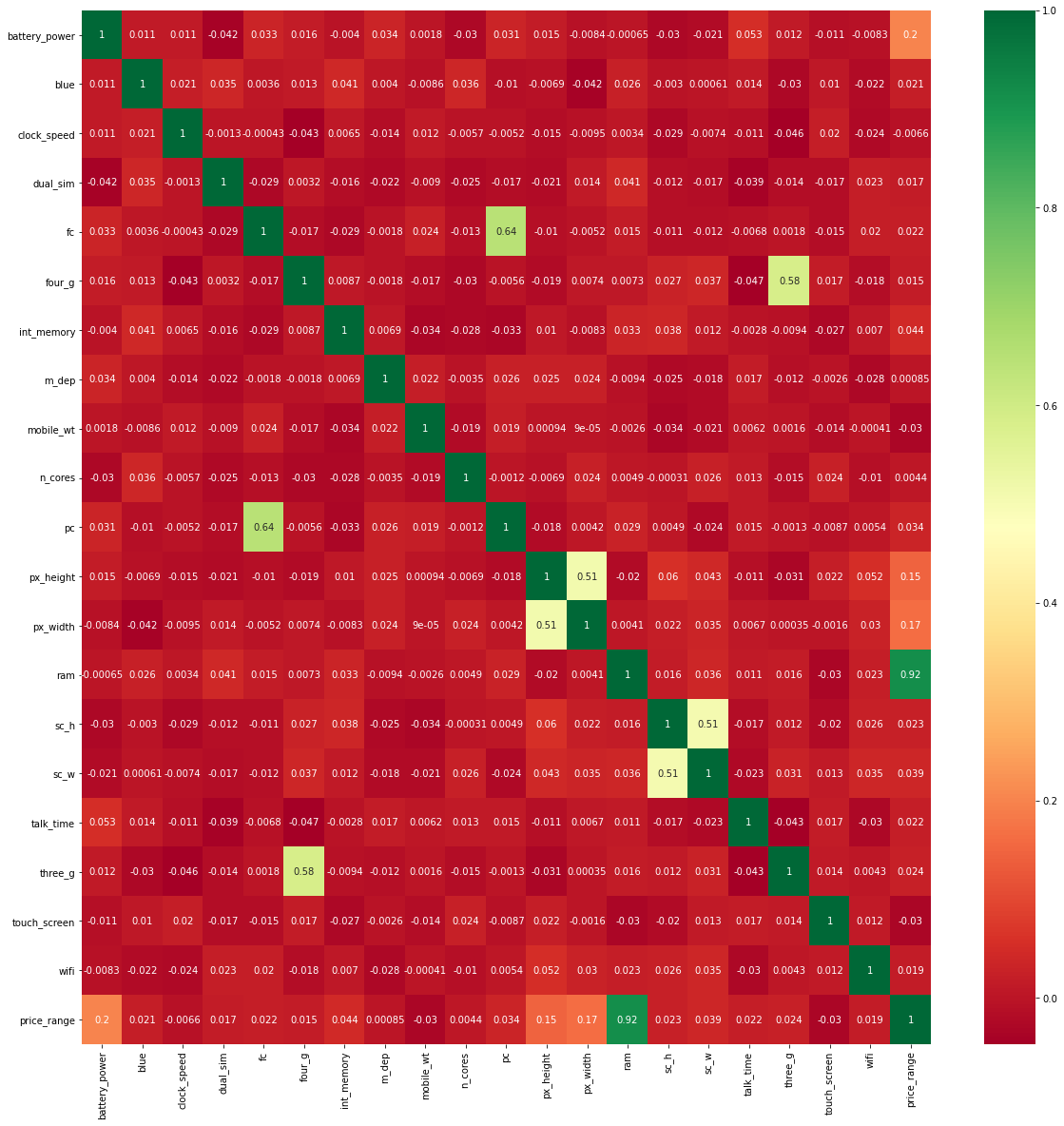
corrmat = mobile\_data.corr()

top\_corr\_features = corrmat.index

plt.figure(figsize=(20,20))

*#plot heat map*

g=sns.heatmap(mobile\_data[top\_corr\_features].corr(),annot=True,cmap="RdYlGn")



Note: From above correlation plot we can conclude that the feature 'price\_range' and 'ram' are highly correlated features which can also be related with the present situation. As ram of your mobile increases price also gets increases.