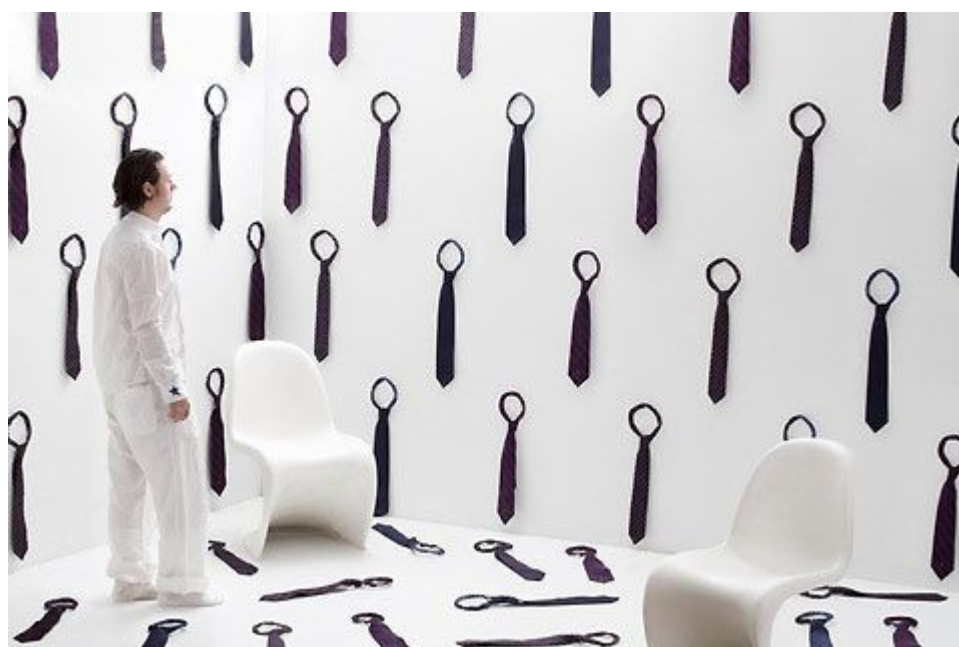


# Feature Selection Techniques in Machine Learning with Python



Raheel Shaikh

*With the new day comes new strength and new thoughts* — **Eleanor Roosevelt**



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

**Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model.** The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

Irrelevant or partially relevant features can negatively impact model performance.

Feature selection and Data cleaning should be the first and most important step of your model designing.

In this post, you will discover feature selection techniques that you can use in Machine Learning.

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

**How to select features and what are Benefits of performing feature selection before modeling your data?**

- **Reduces Overfitting:** Less redundant data means less opportunity to make decisions based on noise.
- **Improves Accuracy:** Less misleading data means modeling accuracy improves.
- **Reduces Training Time:** fewer data points reduce algorithm complexity and algorithms train faster.

**I want to share my personal experience with this.**

I prepared a model by selecting all the features and I got an accuracy of around 65% which is not pretty good for a predictive model and after doing some feature selection and feature engineering without doing any logical changes in my model code my accuracy jumped to 81% which is quite impressive

Now you know why I say feature selection should be the first and most important step of your model design.

**Feature Selection Methods:**

I will share 3 Feature selection techniques that are easy to use and also gives good results.

1. Univariate Selection

2. Feature Importance

3. Correlation Matrix with Heatmap

Let's have a look at these techniques one by one with an example

**You can download the dataset from here**

**<https://www.kaggle.com/iabhishekofficial/mobile-price-classification#train.csv>**

**Description of variables in the above file**

battery\_power: Total energy a battery can store in one time measured in mAh

blue: Has Bluetooth or not

clock\_speed: the speed at which microprocessor executes instructions

dual\_sim: Has dual sim support or not

fc: Front Camera megapixels

four\_g: Has 4G or not

int\_memory: Internal Memory in Gigabytes

m\_dep: Mobile Depth in cm

mobile\_wt: Weight of mobile phone

n\_cores: Number of cores of the processor

pc: Primary Camera megapixels

px\_height

Pixel Resolution Height

px\_width: Pixel Resolution Width

ram: Random Access Memory in MegaBytes

sc\_h: Screen Height of mobile in cm

sc\_w: Screen Width of mobile in cm

talk\_time: the longest time that a single battery charge will last when you are

three\_g: Has 3G or not

touch\_screen: Has touch screen or not

wifi: Has wifi or not

price\_range: This is the target variable with a value of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost).

## 1. Univariate Selection

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 ( $\chi^2$ ) statistical test for non-negative features to select 10 of the best features from the Mobile Price Range Prediction Dataset.

```
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

data = pd.read_csv("D://Blogs//train.csv")
X = data.iloc[:,0:20] #independent columns
y = data.iloc[:,-1]   #target column i.e price range

#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

```
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score'] #naming the dataframe
columns
print(featureScores.nlargest(10,'Score')) #print 10 best features
```

	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

---

Top 10 Best Features using SelectKBest class

## 2. Feature Importance

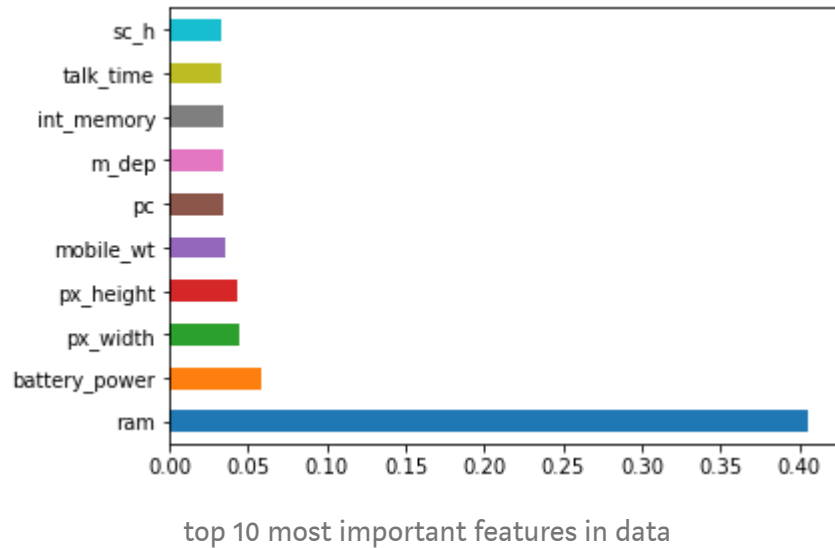
You can get the feature importance of each feature of your dataset by using the feature importance property of the model.

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, we will be using Extra Tree Classifier for extracting the top 10 features for the dataset.

```
import pandas as pd
import numpy as np

data = pd.read_csv("D://Blogs//train.csv")
X = data.iloc[:,0:20] #independent columns
y = data.iloc[:,-1] #target column i.e price range
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_) #use inbuilt class
feature_importances of tree based classifiers
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_,
index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



### 3. Correlation Matrix with Heatmap

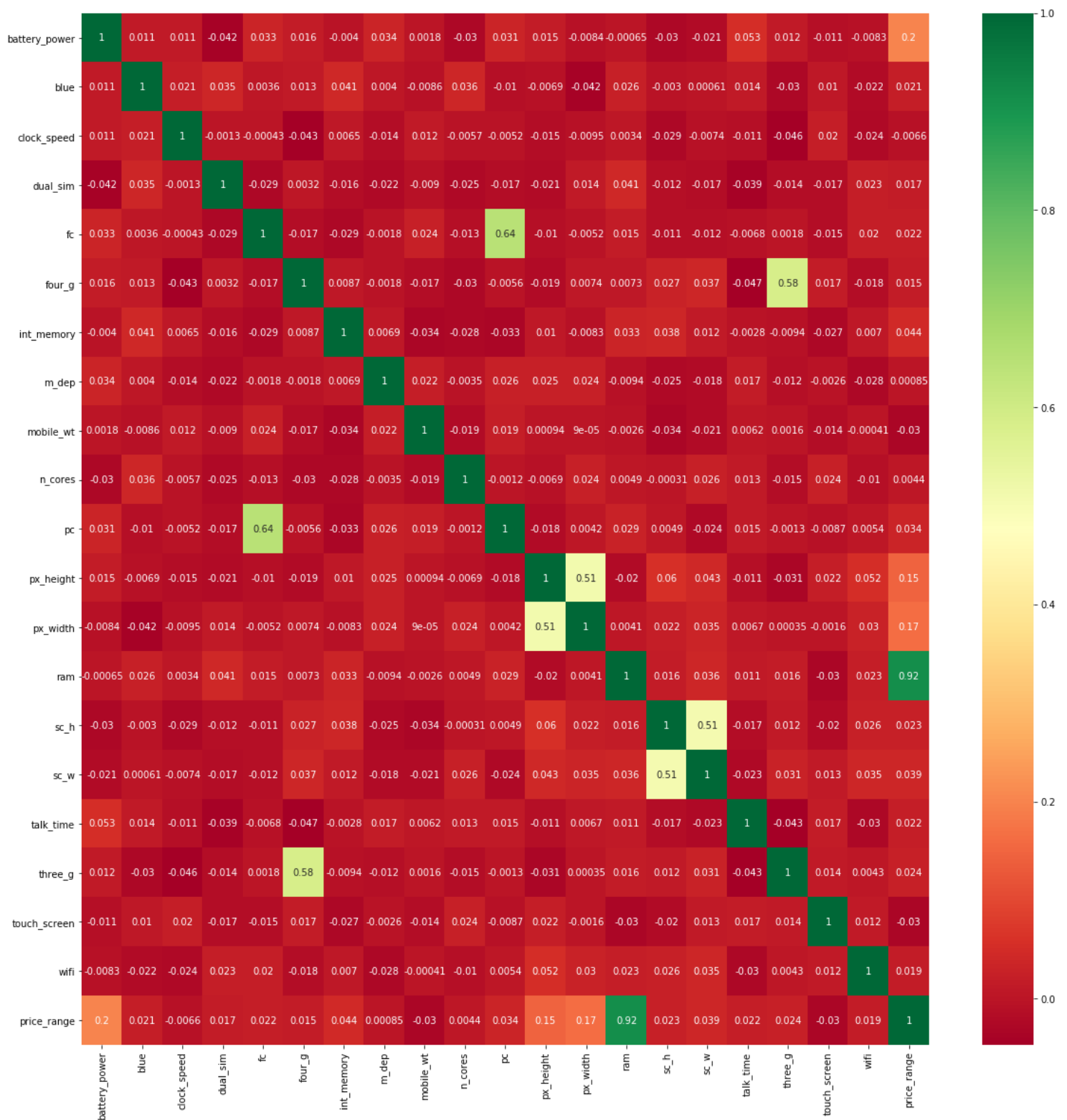
Correlation states how the features are related to each other or the target variable.

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.

```
import pandas as pd
import numpy as np
import seaborn as sns

data = pd.read_csv("D://Blogs//train.csv")
X = data.iloc[:,0:20] #independent columns
y = data.iloc[:,-1]   #target column i.e price range
#get correlations of each features in dataset
corrmat = data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



Have a look at the last row i.e price range, see how the price range is correlated with other features, ram is the highly correlated with price range followed by battery power, pixel height and width while m\_dep, clock\_speed and n\_cores seems to be least correlated with price\_range.