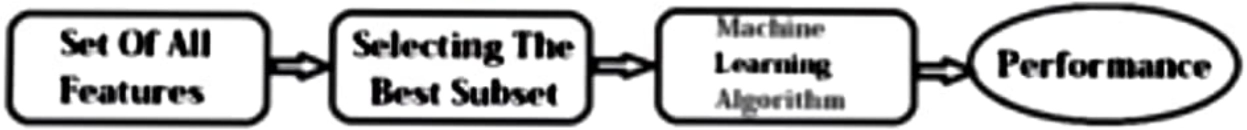
1

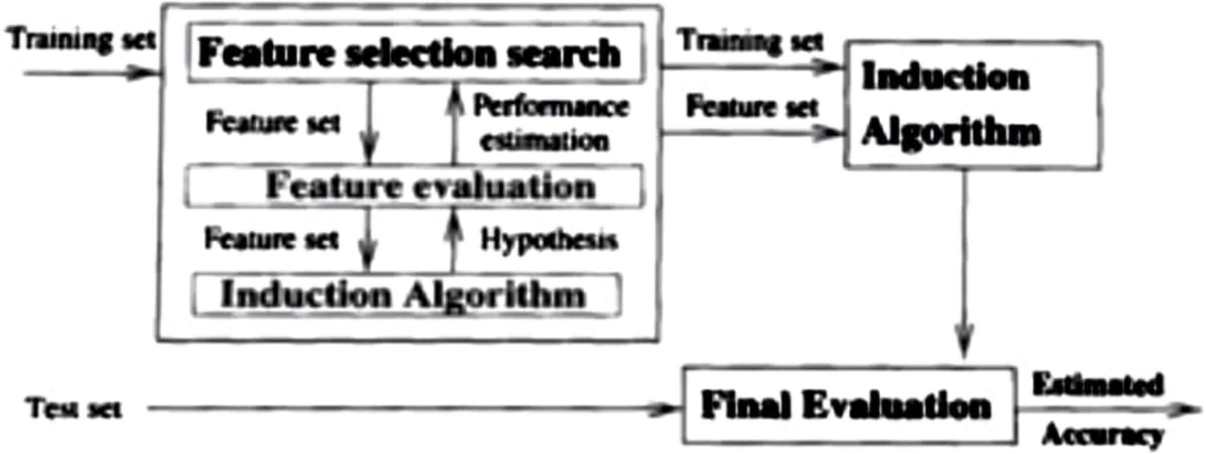
Filter Method

ANOVA



2

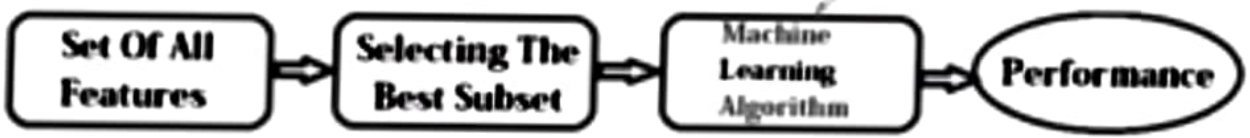
Wrapper Method



1

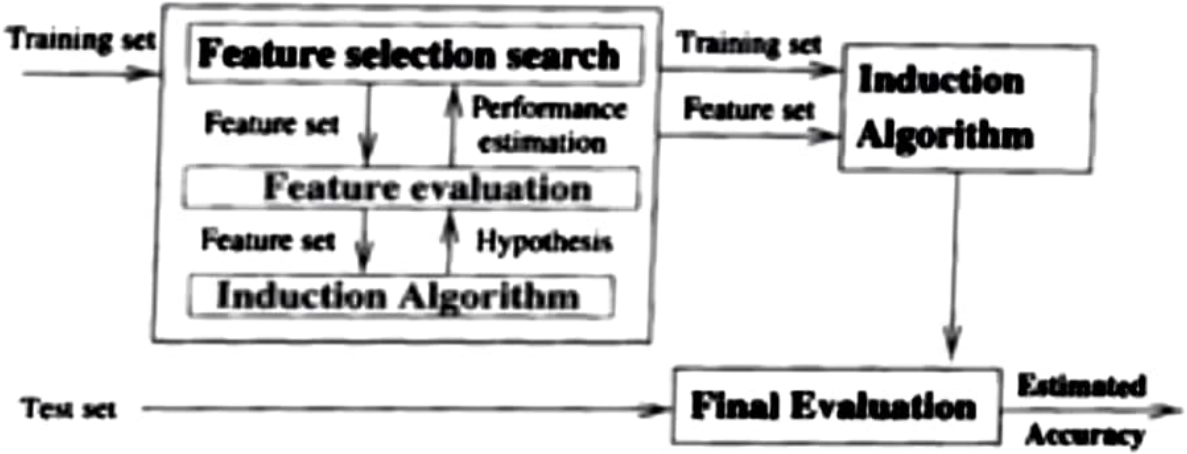
Filter Method

ANOVA TEST



2

Wrapper Method



jupyter **Feature Selection** Last Checkpoint 4 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Markdown

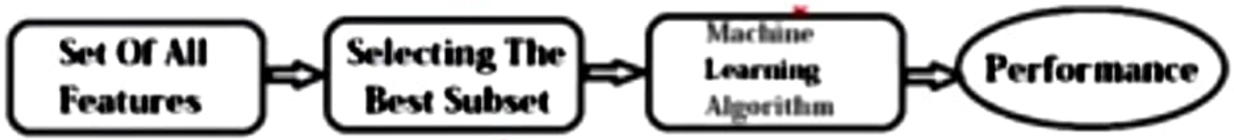
3. Correlation Matrix with Heatmap

CHI SQUARE
ANOVA TEST

1

Filter Method

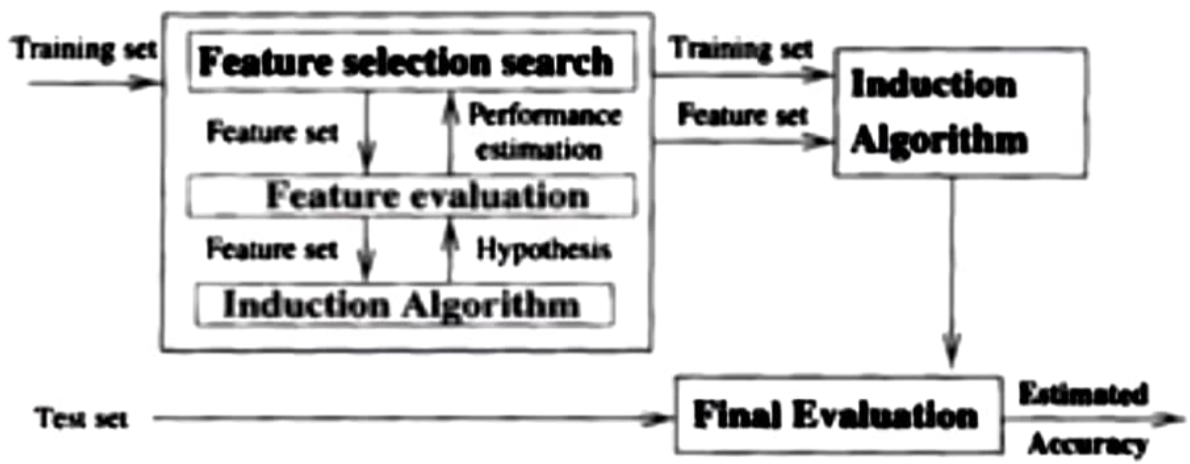
9



Co-relation Coefficient

2

Wrapper Method



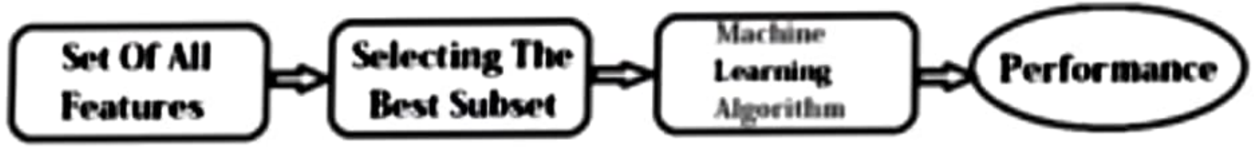
3 Correlation Matrix with Heatmap

X ↔ Y Highly Correlated

X	Y
1	10
2	20
3	30
4	40

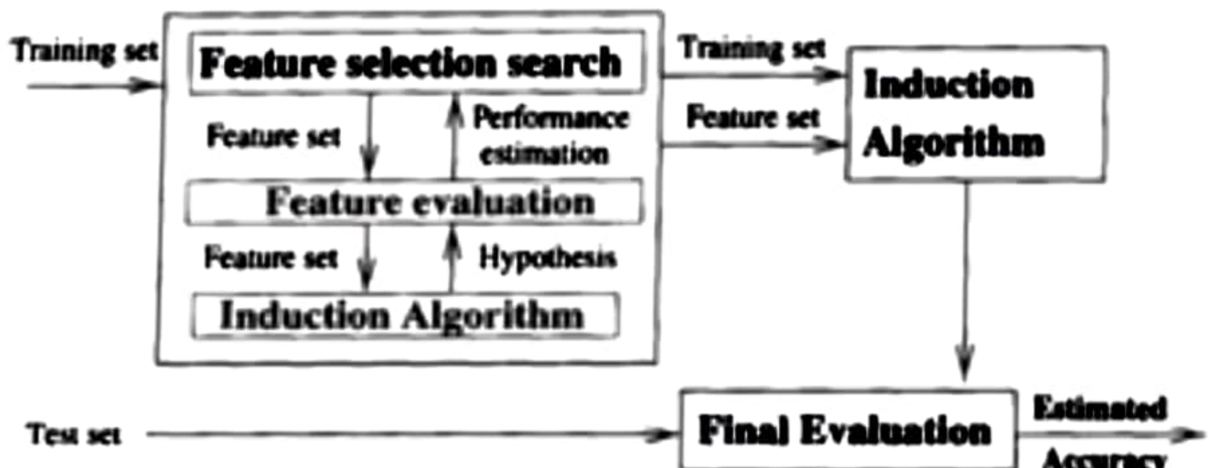
1

Filter Method



2

Wrapper Method



Covariance,

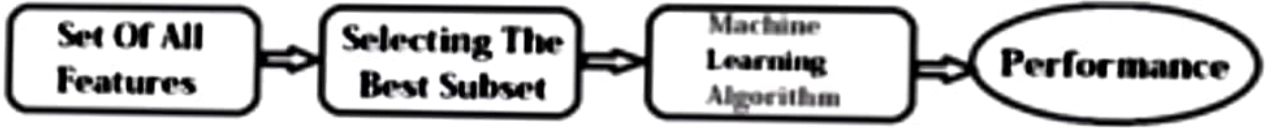
Correlation

$X \leftrightarrow Y$ Highly Correlated

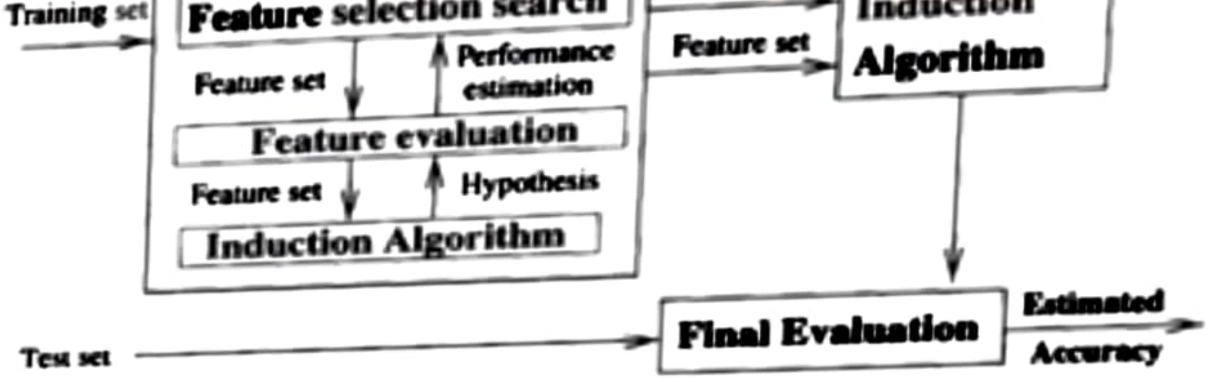
-1 to 1

X	Y
1	10
2	20
3	30
4	40

1 Filter Method



2 Wrapper Method



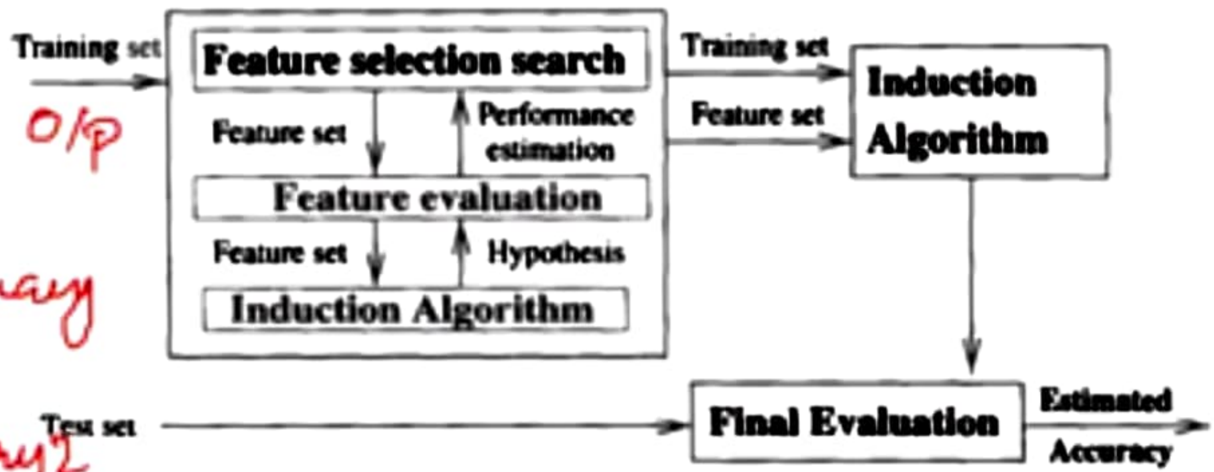
1. **Forward Selection**: Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
2. **Backward Elimination**: 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 Methods

A B C D E O/p

→ Model → Accuracy

→ Model → Accuracy2



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 Methods

A B C D E

O/P

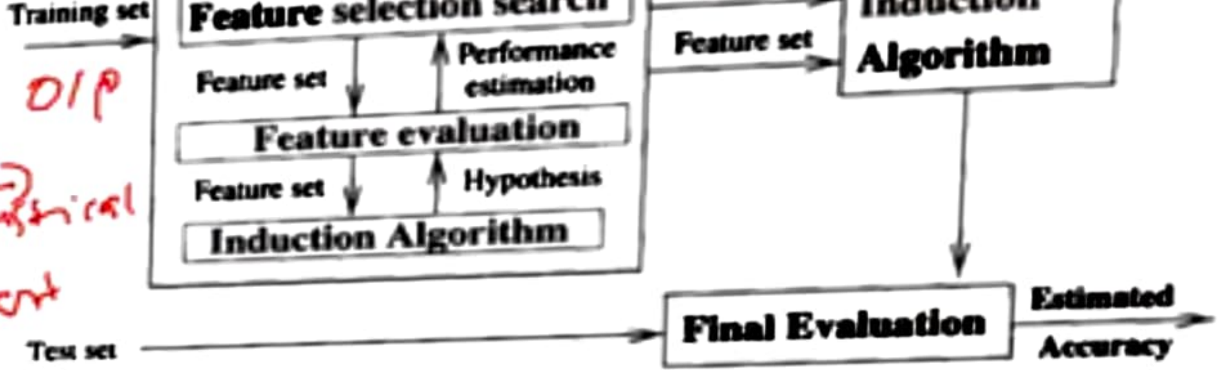
Model

statistical test

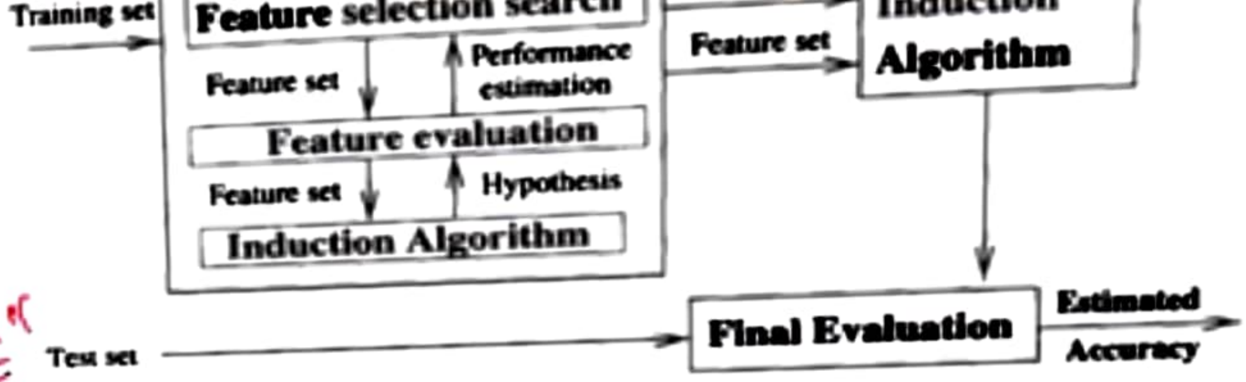
QUARE

0.05

>0.05



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.



dataset

very smaller

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 Methods

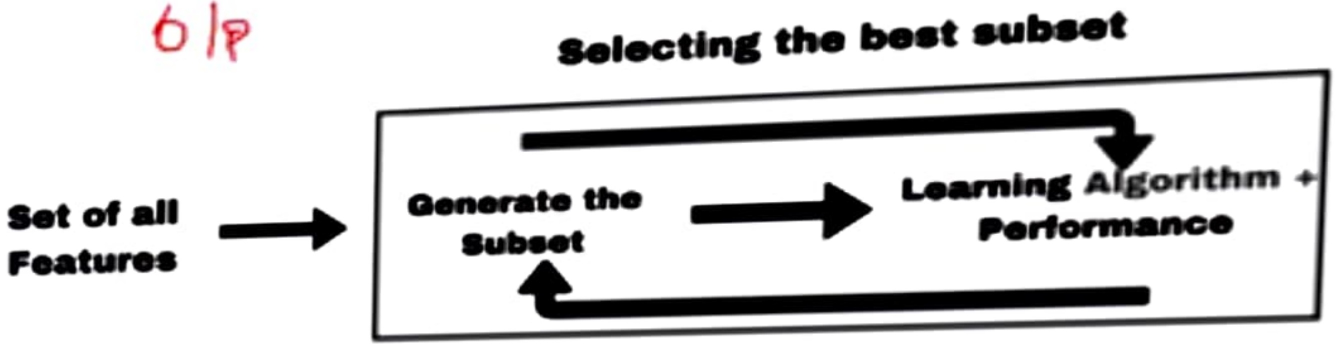
the performance of the model. ...

3. **Recursive Feature elimination**: It is a greedy optimization algorithm ...
and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the ...
exhausted. It then ranks the features based on the order of their elimination.

3. ## Embedded Methods

A B C D E

O/P



Univariate Selection

Statistical tests can be used to select those features that have the strongest relationship with the output variable.

ranks the features based on the order of their elimination ... builds the next model with the left features until all the features are

3

A B C D E

Embedded Methods

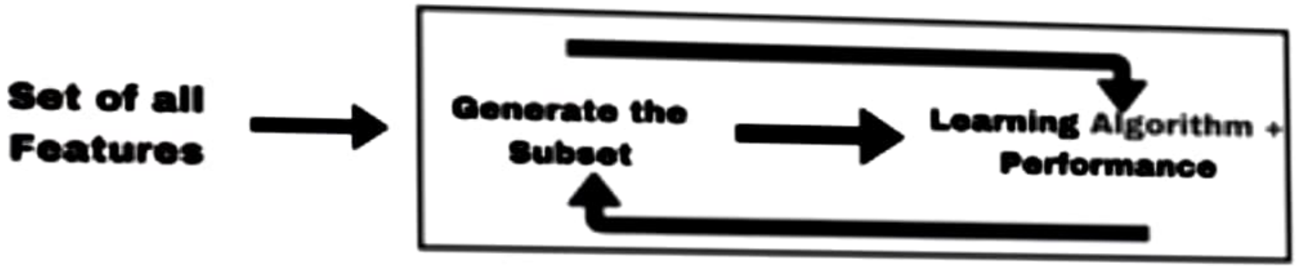
O/P

A → □ → Acc

AB → □ → Acc

B → □ → Acc

Selecting the best subset



Univariate Selection

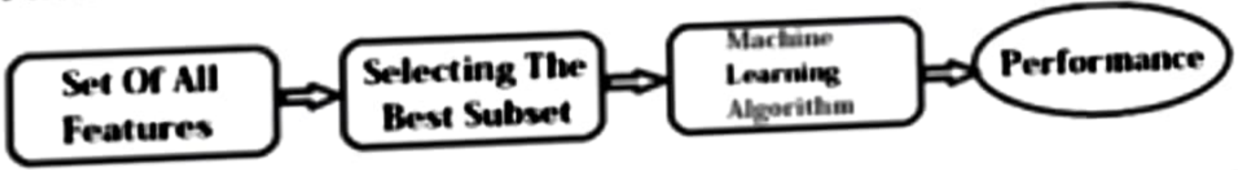
Statistical tests can be used to select those features that have the strongest relationship with the output variable

Feature Selection

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

1. Filter Method



2. Wrapper Method

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.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
data = pd.read_csv("train.csv")
X = data.iloc[:,0:20]  #independent columns
y = data.iloc[:,-1]    #target column i.e price range
```

In [2]:
```
#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
```

In [3]:
```
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

K = 10

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 (chi²) statistical test for non-negative features to select 10 of the best features from the Mobile Price Range Prediction Dataset

2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15

5 rows × 21 columns

In [2]:
```
#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
```

In [3]:
```
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

In [5]:
```
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
```

In [6]:
```
featureScores
```

localhost:8888/notebooks/Medical%20Science/chest_xray/chest_xray/Feature%20Selection.ipynb

jupyter **Feature Selection** Last Checkpoint: 4 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3

In [6]:
```
featureScores
```

Out[6]:

	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

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(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
```

localhost:8888/notebooks/Medical%20Science/chest_xray/chest_xray/Feature%20Selection.ipynb

jupyter Feature Selection Last Checkpoint: 4 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

```
6     int_memory    89.839124
15          sc_w    16.480319
16     talk_time    13.236400
4             fc    10.135166
14          sc_h     9.614878
```

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.

In [9]:
```python
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
```

```
C:\Users\krish.naik\AppData\Local\Continuum\anaconda3\envs\myenv\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarnin
g: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[9]:
```
ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
           max_depth=None, max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

localhost:8888/notebooks/Medical%20Science/chest_xray/chest_xray/Feature%20Selection.ipynb

jupyter Feature Selection Last Checkpoint: 4 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

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.

In [9]:
```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
```

```
C:\Users\krish.naik\AppData\Local\Continuum\anaconda3\envs\myenv\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarnin
g: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[9]:
```
ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
           max_depth=None, max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [10]:
```
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.05851702 0.02017399 0.02964452 0.01554391 0.03512978 0.01980382
 0.03361452 0.03199869 0.03715415 0.03173022 0.03503685 0.04884264
```

localhost:8888/notebooks/Medical%20Science/chest_xray/chest_xray/Feature%20Selection.ipynb

Singam - Kadhal Va... www.upsc.gov.in/ex... Contact Us - How T... Traffic Domination... CP-006 Video - 147... Search - Visual Stud... How to Customize... Top 10 Alg

jupyter Feature Selection Last Checkpoint: 4 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

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

In [9]:

```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
```

```
C:\Users\krish.naik\AppData\Local\Continuum\anaconda3\envs\myenv\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarnin
g: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[9]:

```
ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
           max_depth=None, max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [10]:

```
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.05851702 0.02017399 0.02964452 0.01554391 0.03512978 0.01980382
 0.03361452 0.03199869 0.03715415 0.03173022 0.03503685 0.04884264
 0.05099896 0.39904948 0.03242739 0.03439766 0.0342445  0.01444038
 0.01571493 0.02153659]
```

In [11]:

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
```

localhost:8888/notebooks/Medical%20Science/chest_xray/chest_xray/Feature%20Selection.ipynb

jupyter Feature Selection Last Checkpoint: 4 hours ago (autosaved)

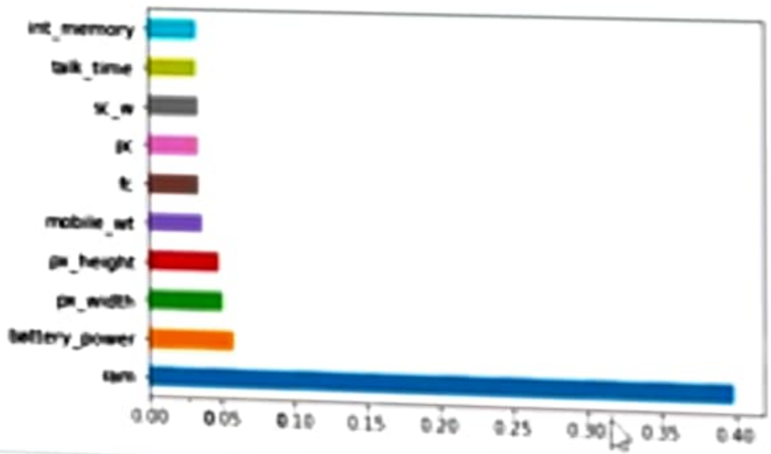
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [10]:
```
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.05851702 0.02017399 0.02964452 0.01554391 0.03512978 0.01980382
 0.03361452 0.03199869 0.03715415 0.03173022 0.03503685 0.04884264
 0.05099896 0.39904948 0.03242739 0.03439766 0.0342445  0.01444038
 0.01571493 0.02153659]
```

In [11]:
```
#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()
```



localhost:8888/notebooks/Medical%20Science/chest_xray/chest_xray/Feature%20Selection.ipynb

jupyter **Feature Selection** Last Checkpoint: 4 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

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 [17]:

```python
import seaborn as sns
#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")
```