Feature_Selection

- **Having irrelevant features in your data can decrease the accuracy of the models and makes your models learn based on irrelevant**

Defination

Feature Selection :

- Process of selecting the best features which contribute maximum for the model in order to get best result in term of accuracy or it should take less time for traning .
- Feature selection methods are intended to reduce the number of input variables to those that are believed to be most useful to a model in order to predict the target variable.

Benefits of Performing Feature-Selection :

```
1.  Reduce Overfitting
2.  Improve Accuracy
3.  Reduce Traning Time
```

In [3]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [4]:
```python
pwd
```

Out[4]:
```
'E:\\DataScience\\MachineLearning\\Breast-cancer-detection-using-ML'
```

In [5]:
```python
path='E:\\DataScience\\MachineLearning\\Breast-cancer-detection-using-ML'
```

In [6]:
```python
import os
os.listdir(path)
```

Out[6]:
```
['.ipynb_checkpoints',
 'Breast_Cancer_Detection_Using_ML.ipynb',
 'Breast_Cancer_Detection_Using_ML.pdf',
 'Breast_Cancer_Detection_Using_ML.py',
 'data.csv',
 'Feature_Selection.ipynb']
```

In [7]:
```python
#reading data
df =pd.read_csv(path+"\\data.csv")
```

In [8]:
```python
df
```

Out[8]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poi
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	
...	...	...	...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	

569 rows × 33 columns

In [9]:
```python
df.columns
```

Out[9]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

In [10]:
```python
df.shape
```

Out[10]:
```
(569, 33)
```

In [11]:
```python
df.isnull().sum()
```

Out[11]:
```
id                           0
diagnosis                    0
radius_mean                  0
texture_mean                 0
perimeter_mean               0
area_mean                    0
smoothness_mean              0
compactness_mean             0
concavity_mean               0
concave points_mean          0
symmetry_mean                0
fractal_dimension_mean       0
radius_se                    0
texture_se                   0
perimeter_se                 0
area_se                      0
smoothness_se                0
compactness_se               0
concavity_se                 0
concave points_se            0
symmetry_se                  0
fractal_dimension_se         0
radius_worst                 0
texture_worst                0
perimeter_worst              0
area_worst                   0
smoothness_worst             0
compactness_worst            0
concavity_worst              0
concave points_worst         0
symmetry_worst               0
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

In [12]:
```python
df=df.drop(['Unnamed: 32','id'],axis=1)
```

In [13]:
```python
df
```

Out[13]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430
...	...	...	...	...	...	...	...	...	...
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200
568	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000

569 rows × 31 columns

Univariate feature selection

For regression: f_regression, mutual_info_regression **For classification**: chi2, f_classif, mutual_info_classif

In [14]:
```python
#importing feauture_selection from sklearn
# stastistical fuction --- chi
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2 ,f_classif

#traning data -- all features except the target values
#here we are using  data shape -- 569 , 32
X =df.iloc[:,1:]
#target values
y =df.iloc[:,0]

# k= 15 : selecting top 15 values  which are highly co-related to taget
# using chi2 function
fs_chi2 =SelectKBest(chi2 ,k=15)
X_chi2 = fs_chi2.fit_transform(X,y)

# k= 15 : selecting top 15 values  which are highly co-related to taget
# using f_classif function
fs_f =SelectKBest(f_classif ,k=15)
X_f_classif = fs_f.fit_transform(X,y)
```

In [15]:
```python
#X_chi2 selected
dfscores = pd.DataFrame(fs_chi2.scores_)
dfcolumns =pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
#naming the dataframe columns
featureScores.columns = ['Specs','Score']
print(featureScores.nlargest(15,'Score'))
```

```
                   Specs          Score
23            area_worst  112598.431564
3              area_mean   53991.655924
13               area_se    8758.504705
22       perimeter_worst    3665.035416
2         perimeter_mean    2011.102864
20          radius_worst     491.689157
0            radius_mean     266.104917
12          perimeter_se     250.571896
21         texture_worst     174.449400
1           texture_mean      93.897508
26       concavity_worst      39.516915
10             radius_se      34.675247
6         concavity_mean      19.712354
25     compactness_worst      19.314922
27  concave points_worst      13.485419
```

In [16]:
```python
#X_f_classif Selected
dfscores = pd.DataFrame(fs_f.scores_)
dfcolumns =pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
#naming the dataframe columns
featureScores.columns = ['Specs','Score']
print(featureScores.nlargest(15,'Score'))
```

```
                   Specs       Score
27  concave points_worst  964.385393
22       perimeter_worst  897.944219
7    concave points_mean  861.676020
20          radius_worst  860.781707
2         perimeter_mean  697.235272
23            area_worst  661.600206
0            radius_mean  646.981021
3              area_mean  573.060747
6         concavity_mean  533.793126
26       concavity_worst  436.691939
5       compactness_mean  313.233079
25     compactness_worst  304.341063
10             radius_se  268.840327
12          perimeter_se  253.897392
13               area_se  243.651586
```

Feature Importance

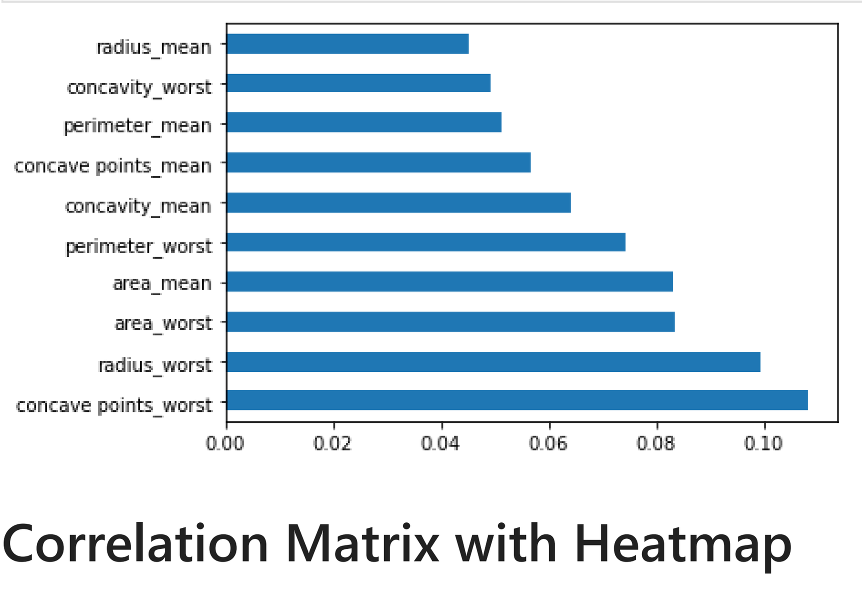
- Feature importance gives you a score for each feature of your data,
- The higher the score more important or relevant is the feature towards your output variable.
- Feature importance is an inbuilt class that comes with Tree Based Classifiers,
- We will be using Extra Tree Classifier for extracting the top 10 features for the dataset.

In [17]:
```python
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
```

In [18]:
```python
model.fit(X,y)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.04517047 0.01606722 0.05111449 0.08301556 0.00874529 0.0233687
 0.06424779 0.05671327 0.00715549 0.00531724 0.01738561 0.00494817
 0.01884134 0.03305324 0.00577992 0.00600593 0.00966862 0.0119452
 0.00600117 0.00729472 0.0995078  0.02793595 0.07422834 0.08355751
 0.01967976 0.02634849 0.04936704 0.10803962 0.01739662 0.01209941]
```

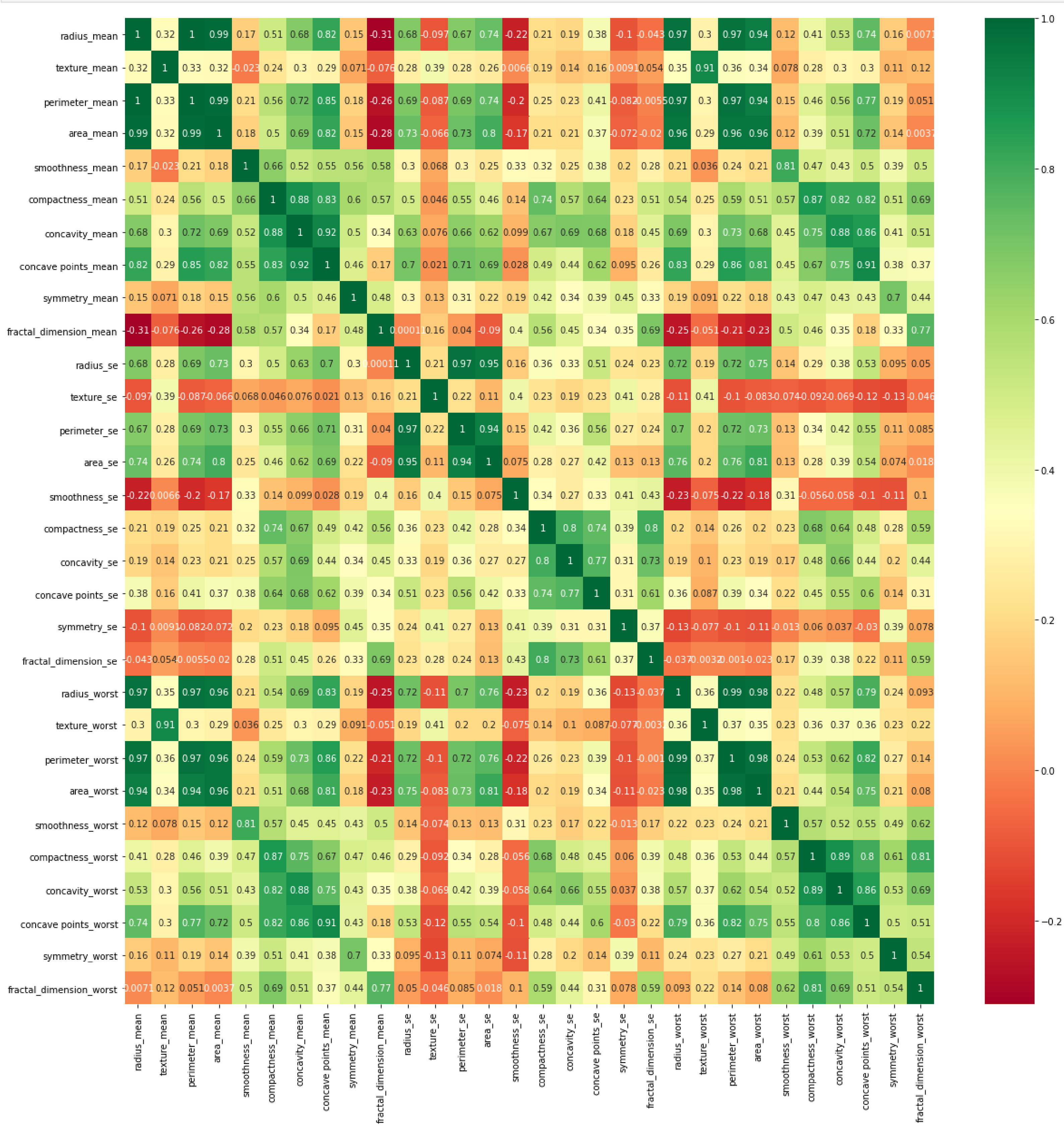
In [19]:
```python
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



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.

In [22]:
```python
#get correlations of each features in dataset
corr = df.corr()
top_corr_features = corr.index
plt.figure(figsize=(20,20))
#plot heat map
plt=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



In [ ]: