Study Case : Winscosin Breast Cancer

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. n the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server: ftp ftp.cs.wisc.edu cd math-prog/cpo-dataset/machine-learn/WDBC/

Also can be found on UCI Machine Learning Repository: [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29 (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29)

Attribute Information:

1) ID number 2) Diagnosis (M = malignant, B = benign) 3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter) b) texture (standard deviation of gray-scale values) c) perimeter d) area e) smoothness (local variation in radius lengths) f) compactness (perimeter^2 / area - 1.0) g) concavity (severity of concave portions of the contour) h) concave points (number of concave portions of the contour) i) symmetry j) fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits.

Missing attribute values: none

Class distribution: 357 benign, 212 malignant

Import Library and Dataset

In [1]:
```
import pandas as pd
import pandas_profiling
import numpy as np
import matplotlib.pyplot as plt
% matplotlib inline
import seaborn as sns
sns.set()
plt.style.use('bmh')
```

In [2]:
```
df=pd.read_csv('Dataset/data.csv')
print("Dataset size : ",df.shape)
df=df.drop(columns=['id','Unnamed: 32'])
df.head()
```

```
Dataset size :  (569, 33)
```

Out[2]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	M	17.99	10.38	122.80	1001.0	0.11840
1	M	20.57	17.77	132.90	1326.0	0.08474
2	M	19.69	21.25	130.00	1203.0	0.10960
3	M	11.42	20.38	77.58	386.1	0.14250
4	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 31 columns

In [3]:
```
pandas_profile=pandas_profiling.ProfileReport(df)
pandas_profile.to_file(outputfile='Pandas_ProfilingOutput.html')
#pandas_profile
```

Detail HTML Pandas Profiling (Pandas_ProfilingOutput.html)

Explore the Values

Explore Distribution values from the dataset using describe statistic and histogram

In [4]:
```
df.describe()
```

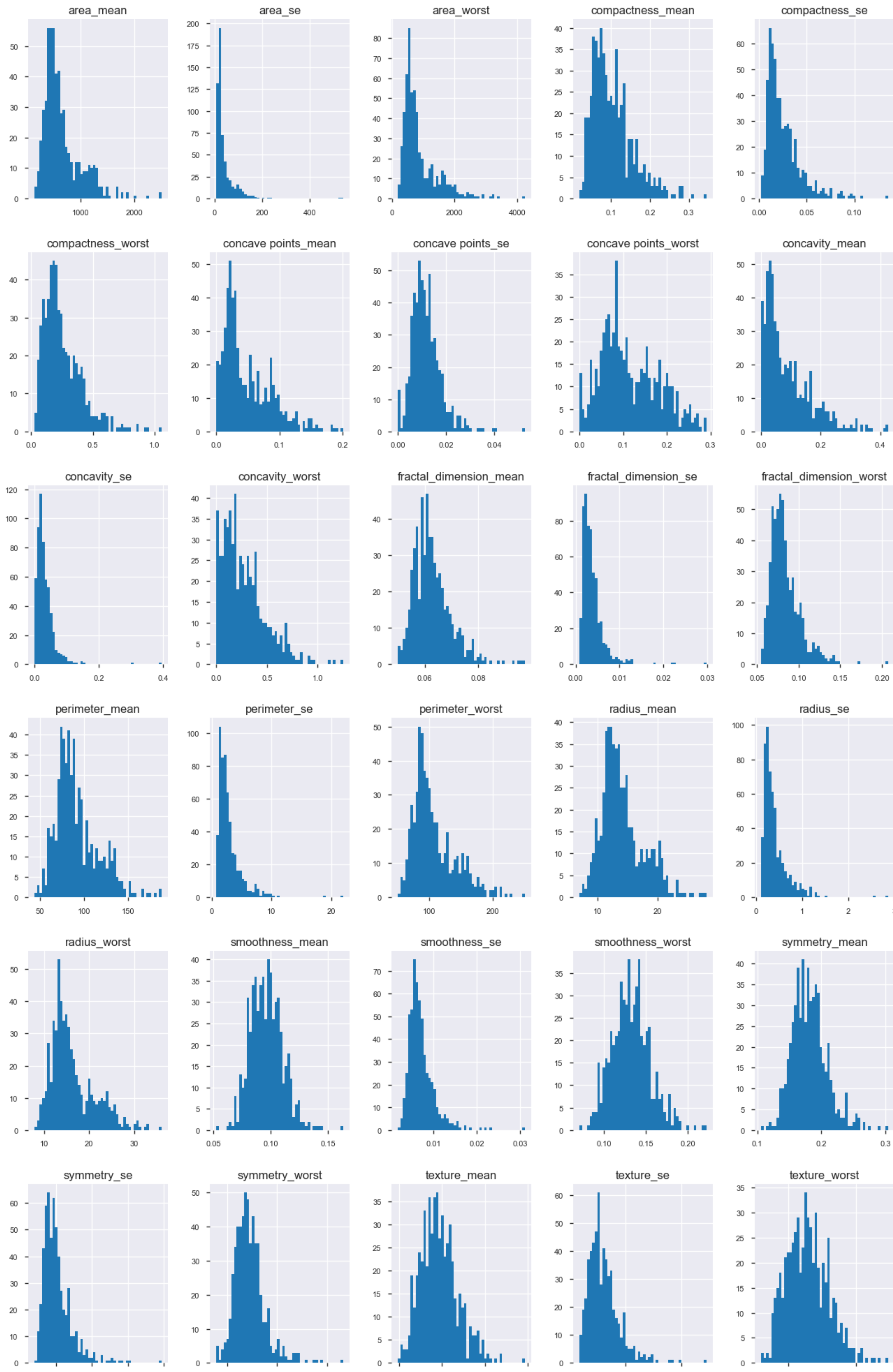
Out[4]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
count	569.000000	569.000000	569.000000	569.000000	569.000000	5
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.

8 rows × 30 columns

In [5]:

```
df.hist(figsize=(16,25),bins=50,xlabelsize=8,ylabelsize=8);
```



Training Dataset Preparation

Since most of the Algorithm machine learning only accept array like as input, so we need to create an array from dataframe set to X and y array before running machine learning algorithm

In [6]:
```
X=np.array(df.drop(columns=['diagnosis']))
y=df['diagnosis'].values
```

In [7]:
```
print ("X dataset shape : ",X.shape)
print ("y dataset shape : ",y.shape)
```

```
X dataset shape :  (569, 30)
y dataset shape :  (569,)
```

The dataset is splitted by X the parameter and y for classification labels

Machine Learning Model

Import Machine Learning Library from Scikit-Learn

In [8]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

Using 5 Machine Learning Model

Machine learning model used is Classification model, since the purpose of this Study case is to classify diagnosis between "Malignant" (M) Breast Cancer and "Benign" (B) Breast Cancer

- Model 1 : Using Simple Logistic Regression
- Model 2 : Using Support Vector Classifier
- Model 3 : Using Decision Tree Classifier
- Model 4 : Using Random Forest Classifier
- Model 5 : Using Gradient Boosting Classifier

In [9]:
```
model_1 = LogisticRegression()
model_2 = SVC()
model_3 = DecisionTreeClassifier()
model_4 = RandomForestClassifier()
model_5 = GradientBoostingClassifier()
```

Model Fitting

since we need to fit the dataset into algorithm, so proper spliting dataset into training set and test set are required

Method 1. Train test split

Using Scikit learn built in tools to split data into training set and test set to check the result score of the model train_test_split configuration using 20% data to test and 80& data to train the model, random_state generator is 45.

In [10]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=45)

print ("Train size : ",X_train.shape)
print ("Test size : ",X_test.shape)
```

```
Train size :  (455, 30)
Test size :  (114, 30)
```

Fitting train dataset into model

In [11]:
```
model_1.fit(X_train,y_train)
model_2.fit(X_train,y_train)
model_3.fit(X_train,y_train)
model_4.fit(X_train,y_train)
model_5.fit(X_train,y_train)
```

Out[11]:
```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, 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=100,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm_start=False)
```

Predict and show Score and F1 Score prediction using test data

In [12]:
```
# Predict data
y_pred1=model_1.predict(X_test)
y_pred2=model_2.predict(X_test)
y_pred3=model_3.predict(X_test)
y_pred4=model_4.predict(X_test)
y_pred5=model_5.predict(X_test)

#Show F1 Score
from sklearn.metrics import f1_score
f1_model1=f1_score(y_test,y_pred1,average='weighted',labels=np.unique(y_pred1
))
f1_model2=f1_score(y_test,y_pred2,average='weighted',labels=np.unique(y_pred2
))
f1_model3=f1_score(y_test,y_pred3,average='weighted',labels=np.unique(y_pred3
))
f1_model4=f1_score(y_test,y_pred4,average='weighted',labels=np.unique(y_pred4
))
f1_model5=f1_score(y_test,y_pred5,average='weighted',labels=np.unique(y_pred5
))

print("F1 score Model 1 : ",f1_model1)
print("F1 score Model 2 : ",f1_model2)
print("F1 score Model 3 : ",f1_model3)
print("F1 score Model 4 : ",f1_model4)
print("F1 score Model 5 : ",f1_model5)
```

```
F1 score Model 1 :  0.9557756825927252
F1 score Model 2 :  0.7741935483870968
F1 score Model 3 :  0.9380859556298152
F1 score Model 4 :  0.9734654095556352
F1 score Model 5 :  0.9734654095556352
```

Method 2. Cross validation method

Using Cross validation will resulted in more reliability of the model
in this case using StratifiedKFold from Scikit Learn, with n_split = 10 times and Shuffle = True

In [14]:
```
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=10, shuffle=True)
skf.get_n_splits(X,y)
```

Out[14]:
```
10
```

In [15]:
```
# Set Container to gather the cross validation result of the model
score_list_model1,score_list_model2,score_list_model3,score_list_model4,score_
list_model5 = [],[],[],[],[]
```

In [16]:
```
for train_index, test_index in skf.split(X,y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    model_1.fit(X_train, y_train)
    model_2.fit(X_train, y_train)
    model_3.fit(X_train, y_train)
    model_4.fit(X_train, y_train)
    model_5.fit(X_train, y_train)
    y_pred1=model_1.predict(X_test)
    y_pred2=model_2.predict(X_test)
    y_pred3=model_3.predict(X_test)
    y_pred4=model_4.predict(X_test)
    y_pred5=model_5.predict(X_test)
    score_list_model1.append(f1_score(y_test,y_pred1,average='weighted',labels
=np.unique(y_pred1)))
    score_list_model2.append(f1_score(y_test,y_pred2,average='weighted',labels
=np.unique(y_pred2)))
    score_list_model3.append(f1_score(y_test,y_pred3,average='weighted',labels
=np.unique(y_pred3)))
    score_list_model4.append(f1_score(y_test,y_pred4,average='weighted',labels
=np.unique(y_pred4)))
    score_list_model5.append(f1_score(y_test,y_pred5,average='weighted',labels
=np.unique(y_pred5)))
```

In [17]:
```
score_table = pd.DataFrame({"F1 Score model 1" :score_list_model1,
                           "F1 Score model 2" :score_list_model2,
                           "F1 Score model 3" :score_list_model3,
                           "F1 Score model 4" :score_list_model4,
                           "F1 Score model 5" :score_list_model5})
score_table
```

Out[17]:

	F1 Score model 1	F1 Score model 2	F1 Score model 3	F1 Score model 4	F1 Score model 5
0	1.000000	0.765957	0.948486	0.982676	0.982676
1	0.931549	0.765957	0.847121	0.948486	0.965517
2	0.929018	0.774194	0.929018	0.982362	0.964912
3	0.891967	0.774194	0.910661	0.946397	0.946397
4	0.982362	0.774194	0.858037	0.929018	0.964912
5	0.964912	0.774194	0.895625	0.947610	0.982537
6	0.947610	0.774194	0.947610	0.964509	0.964509
7	0.982051	0.769231	0.946153	1.000000	0.964286
8	0.982051	0.769231	0.927778	1.000000	0.982221
9	0.928571	0.769231	0.928571	0.926743	0.946153

In [18]:
```
final_1=np.mean(score_list_model1)
final_2=np.mean(score_list_model2)
final_3=np.mean(score_list_model3)
final_4=np.mean(score_list_model4)
final_5=np.mean(score_list_model5)

print("F1 Score Average Model_1",final_1)
print("F1 Score Average Model_2",final_2)
print("F1 Score Average Model_3",final_3)
print("F1 Score Average Model_4",final_4)
print("F1 Score Average Model_5",final_5)
```

```
F1 Score Average Model_1 0.9540091280680972
F1 Score Average Model_2 0.7710574943244813
F1 Score Average Model_3 0.9139061230761165
F1 Score Average Model_4 0.9627801708297762
F1 Score Average Model_5 0.9664119969534308
```

Conclusion

After Testing 5 Model of machine learning classifier and testing both using train test split and cross validation method, conclude that **Model 5** which is **Gradient Boosting** winc with crossvalidation F1 Score **0.9621**