Using the Wisconsin breast cancer diagnostic data set for predictive analysis

Buddhini Waidyawansa (12-03-2016)

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.

For this analysis, as a guide to predictive analysis I followed the instructions and discussion on "A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python)" at Analytics Vidhya.

Load Libraries

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_c
sv)

# keeps the plots in one place. calls image as static pngs
%matplotlib inline
import matplotlib.pyplot as plt # side-stepping mpl backend
import matplotlib.gridspec as gridspec # subplots
import mpld3 as mpl

#Import models from scikit learn module:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import KFold   #For K-fold cross vali
dation
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics
```

```
/opt/conda/lib/python3.5/site-packages/sklearn/cross_validation.p
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/cross_validation.p
y:44: DeprecationWarning: This module was deprecated in version 0.
18 in favor of the model_selection module into which all the refac
tored classes and functions are moved. Also note that the interfac
e of the new CV iterators are different from that of this module.
This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Load the data

In [2]:

```
df = pd.read_csv("../input/data.csv",header = 0)
df.head()
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothl
0	842302	M	17.99	10.38	122.80	1001.0	0.1184
1	842517	M	20.57	17.77	132.90	1326.0	0.0847
2	84300903	M	19.69	21.25	130.00	1203.0	0.1096
3	84348301	M	11.42	20.38	77.58	386.1	0.1425
4	84358402	M	20.29	14.34	135.10	1297.0	0.1003

5 rows × 33 columns

Clean and prepare data

In [3]:

```
df.drop('id',axis=1,inplace=True)
df.drop('Unnamed: 32',axis=1,inplace=True)
# size of the dataframe
len(df)
```

Out[3]:

```
569
```

In [4]:

```
df.diagnosis.unique()
```

Out[4]:

```
array(['M', 'B'], dtype=object)
```

In [5]:

```
df['diagnosis'] = df['diagnosis'].map({'M':1,'B':0})
df.head()
```

Out[5]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	1	17.99	10.38	122.80	1001.0	0.11840

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

5 rows × 31 columns

Explore data

In [6]:

```
df.describe()
```

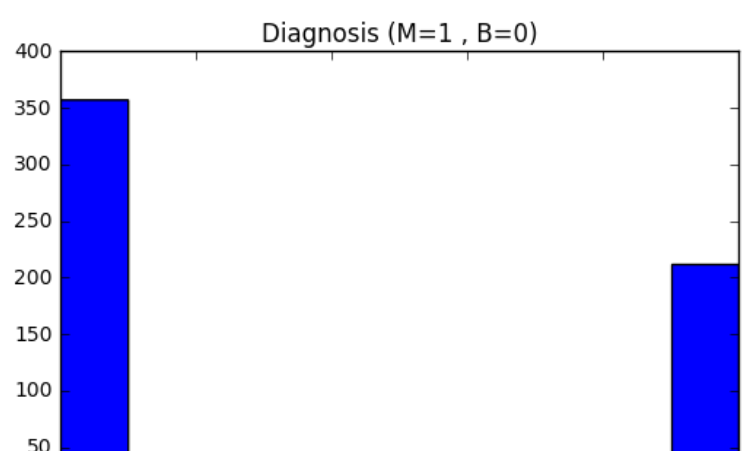
Out[6]:

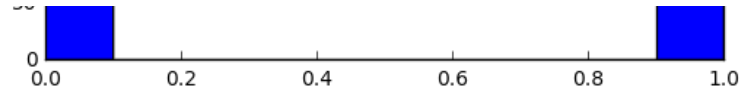
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.00000
mean	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360
std	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064
min	0.000000	6.981000	9.710000	43.790000	143.500000	0.052630
25%	0.000000	11.700000	16.170000	75.170000	420.300000	0.086370
50%	0.000000	13.370000	18.840000	86.240000	551.100000	0.095870
75%	1.000000	15.780000	21.800000	104.100000	782.700000	0.105300
max	1.000000	28.110000	39.280000	188.500000	2501.000000	0.163400

8 rows × 31 columns

In [7]:

```
df.describe()
plt.hist(df['diagnosis'])
plt.title('Diagnosis (M=1 , B=0)')
plt.show()
```





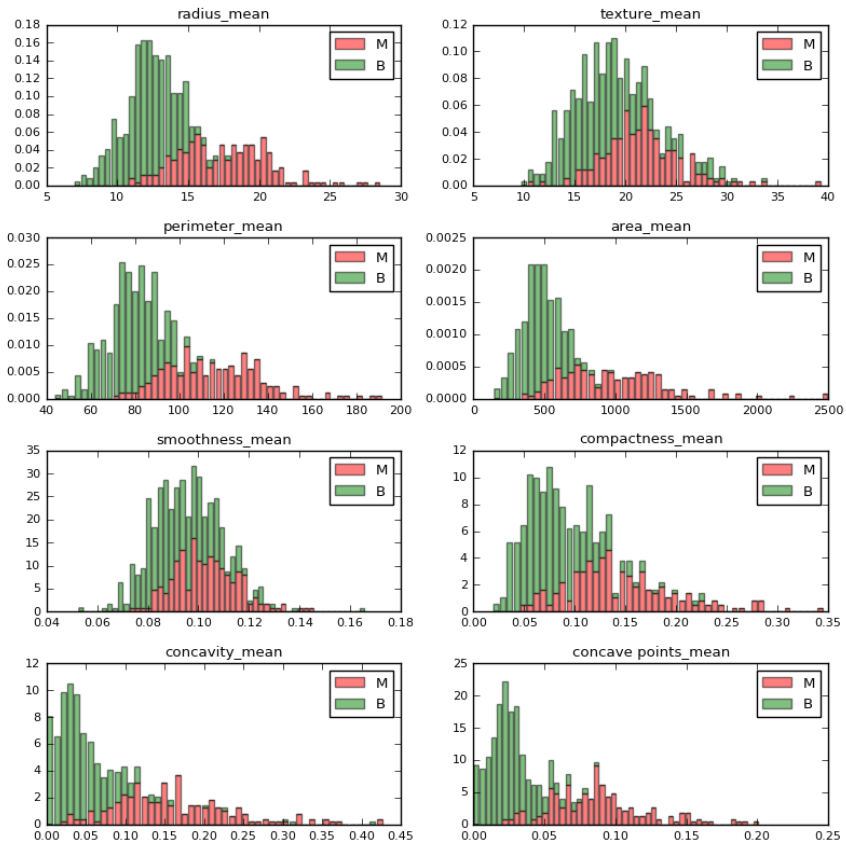
nucleus features vs diagnosis

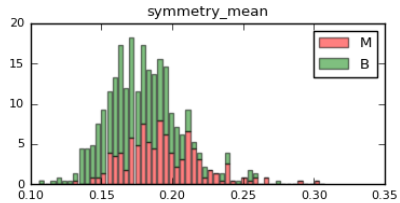
In [8]:

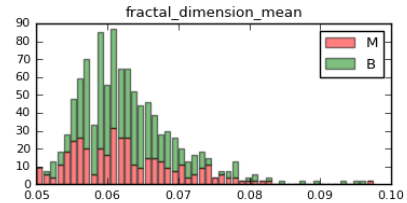
```
features_mean=list(df.columns[1:11])
# split dataframe into two based on diagnosis
dfM=df[df['diagnosis'] ==1]
dfB=df[df['diagnosis'] ==0]
```

In [9]:

```
#Stack the data
plt.rcParams.update({'font.size': 8})
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(8,10))
axes = axes.ravel()
for idx,ax in enumerate(axes):
    ax.figure
    binwidth= (max(df[features_mean[idx]]) - min(df[features_mean[
idx]]))/50
    ax.hist([dfM[features_mean[idx]],dfB[features_mean[idx]]], bin
s=np.arange(min(df[features_mean[idx]]), max(df[features_mean[idx
]]) + binwidth, binwidth) , alpha=0.5,stacked=True, normed = True,
 label=['M','B'],color=['r','g'])
    ax.legend(loc='upper right')
    ax.set_title(features_mean[idx])
plt.tight_layout()
plt.show()
```







Observations

1. mean values of cell radius, perimeter, area, compactness, concavity and concave points can be used in classification of the cancer. Larger values of these parameters tends to show a correlation with malignant tumors.
2. mean values of texture, smoothness, symmetry or fractual dimension does not show a particular preference of one diagnosis over the other. In any of the histograms there are no noticeable large outliers that warrants further cleanup.

Creating a test set and a training set

Since this data set is not ordered, I am going to do a simple 70:30 split to create a training data set and a test data set.

In [10]:

```
traindf, testdf = train_test_split(df, test_size = 0.3)
```

Model Classification

Here we are going to build a classification model and evaluate its performance using the training set.

In [11]:

```
#Generic function for making a classification model and accessing th
e performance.
# From AnalyticsVidhya tutorial
def classification_model(model, data, predictors, outcome):
  #Fit the model:
  model.fit(data[predictors],data[outcome])

  #Make predictions on training set:
  predictions = model.predict(data[predictors])

  #Print accuracy
  accuracy = metrics.accuracy_score(predictions,data[outcome])
  print("Accuracy : %s" % "{0:.3%}".format(accuracy))

  #Perform k-fold cross-validation with 5 folds
  kf = KFold(data.shape[0], n_folds=5)
  error = []
  for train, test in kf:
    # Filter training data
    train_predictors = (data[predictors].iloc[train,:])

    # The target we're using to train the algorithm.
    train_target = data[outcome].iloc[train]
```

```
    # Training the algorithm using the predictors and target.
    model.fit(train_predictors, train_target)

    #Record error from each cross-validation run
    error.append(model.score(data[predictors].iloc[test,:], data[o
utcome].iloc[test]))

    print("Cross-Validation Score : %s" % "{0:.3%}".format(np.mean
(error)))

  #Fit the model again so that it can be refered outside the functio
n:
  model.fit(data[predictors],data[outcome])
```

Logistic Regression model

Logistic regression is widely used for classification of discrete data. In this case we will use it for binary (1,0) classification.

Based on the observations in the histogram plots, we can reasonably hypothesize that the cancer diagnosis depends on the mean cell radius, mean perimeter, mean area, mean compactness, mean concavity and mean concave points. We can then perform a logistic regression analysis using those features as follows:

In [12]:

```
predictor_var = ['radius_mean','perimeter_mean','area_mean','compa
ctness_mean','concave points_mean']
outcome_var='diagnosis'
model=LogisticRegression()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 88.442%
Cross-Validation Score : 88.750%
Cross-Validation Score : 87.500%
Cross-Validation Score : 87.917%
Cross-Validation Score : 87.773%
Cross-Validation Score : 88.193%
```

The prediction accuracy is reasonable. What happens if we use just one predictor? Use the mean_radius:

In [13]:

```
predictor_var = ['radius_mean']
model=LogisticRegression()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 87.940%
Cross-Validation Score : 90.000%
Cross-Validation Score : 88.750%
Cross-Validation Score : 88.333%
Cross-Validation Score : 88.718%
Cross-Validation Score : 87.684%
```

This gives a similar prediction accuracy and a cross-validation score.

The accuracy of the predictions are good but not great. The cross-validation scores are reasonable. Can we do better with another model?

Decision Tree Model

In [14]:

```
predictor_var = ['radius_mean','perimeter_mean','area_mean','compa
ctness_mean','concave points_mean']
model = DecisionTreeClassifier()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 100.000%
Cross-Validation Score : 87.500%
Cross-Validation Score : 87.500%
Cross-Validation Score : 85.000%
Cross-Validation Score : 84.636%
Cross-Validation Score : 84.924%
```

Here we are over-fitting the model probably due to the large number of predictors. Let use a single predictor, the obvious one is the radius of the cell.

In [15]:

```
predictor_var = ['radius_mean']
model = DecisionTreeClassifier()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 97.236%
Cross-Validation Score : 85.000%
Cross-Validation Score : 82.500%
Cross-Validation Score : 83.750%
Cross-Validation Score : 84.015%
Cross-Validation Score : 83.921%
```

The accuracy of the prediction is much much better here. But does it depend on the predictor?

Using a single predictor gives a 97% prediction accuracy for this model but the cross-validation score is not that great.

Randome Forest

In [16]:

```
# Use all the features of the nucleus
predictor_var = features_mean
model = RandomForestClassifier(n_estimators=100,min_samples_split=
25, max_depth=7, max_features=2)
```

```
25, max_depth=7, max_features=2)
classification_model(model, traindf,predictor_var,outcome_var)
```

```
Accuracy : 94.724%
Cross-Validation Score : 93.750%
Cross-Validation Score : 92.500%
Cross-Validation Score : 91.250%
Cross-Validation Score : 90.906%
Cross-Validation Score : 90.953%
```

Using all the features improves the prediction accuracy and the cross-validation score is great.

An advantage with Random Forest is that it returns a feature importance matrix which can be used to select features. So lets select the top 5 features and use them as predictors.

In [17]:

```
#Create a series with feature importances:
featimp = pd.Series(model.feature_importances_, index=predictor_va
r).sort_values(ascending=False)
print(featimp)
```

```
perimeter_mean            0.204561
concave points_mean       0.197067
concavity_mean            0.182823
area_mean                 0.173446
radius_mean               0.115543
compactness_mean          0.043632
texture_mean              0.038492
smoothness_mean           0.019023
symmetry_mean             0.016371
fractal_dimension_mean    0.009043
dtype: float64
```

In [18]:

```
# Using top 5 features
predictor_var = ['concave points_mean','area_mean','radius_mean',
'perimeter_mean','concavity_mean',]
model = RandomForestClassifier(n_estimators=100, min_samples_split
=25, max_depth=7, max_features=2)
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 94.221%
Cross-Validation Score : 92.500%
Cross-Validation Score : 91.875%
Cross-Validation Score : 90.833%
Cross-Validation Score : 90.277%
Cross-Validation Score : 90.449%
```

Using the top 5 features only changes the prediction accuracy a bit but I think we get a better result if we use all the predictors.

What happens if we use a single predictor as before? Just check.

In [19]:

```
predictor_var =  ['radius_mean']
model = RandomForestClassifier(n_estimators=100)
classification_model(model, train
```

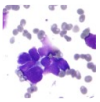


```
Accuracy : 97.236%
```

Data

Data Sources

- Breast Cancer Wiscon...
  - dat... 32 columns



Breast Cancer Wisconsin (Diagnostic) Data Set

Predict whether the cancer is benign or malignant

Last Updated: 2 years ago (Version 2)

About this Dataset

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

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.

Run Info

Succeeded	True	Run Time	613 seconds
Exit Code	0	Queue Time	0 seconds
Docker Image Name	kaggle/python(Dockerfile)	Output Size	0
Timeout Exceeded	False	Used All Space	False
Failure Message			

Log

Download Log


```
Time  Line #  Log Message
           1  [{
           2    "data": "[NbConvertApp] Converting notebook __notebook_source__.ipynb to html\n",
           3    "stream_name": "stderr",
           4    "time": 1.4526910279964795
           5  },{
           6    "data": "[NbConvertApp] Support files will be in __results___files/\n[NbConvertApp] Making directory __results___files\n",
           7    "stream_name": "stderr",
           8    "time": 1.6582987519941526
           9  },{
          10    "data": "[NbConvertApp] Making directory __results___files\n[NbConvertApp] Making directory __results___files\n[NbConvertApp] Making directory __results___files\n[NbConvertApp] Writing 310275 bytes to __results__.html\n",
          11    "stream_name": "stderr",
          12    "time": 1.6628044730023248
          13  }{
          14    "data": "[NbConvertApp] Converting notebook __notebook_source__.ipynb to notebook\n",
          15    "stream_name": "stderr",
          16    "time": 1.5443806919938652
          17  },{
          18    "data": "[NbConvertApp] Executing notebook with kernel: python3\n",
          19    "stream_name": "stderr",
          20    "time": 1.5945803399954457
          21  },{
          22    "data": "[NbConvertApp] Writing 159572 bytes to __notebook__.ipynb\n",
          23    "stream_name": "stderr",
          24    "time": 16.67117001899169
          25  }{
          26    "data": "[NbConvertApp] Converting notebook __notebook__.ipynb to html\n",
          27    "stream_name": "stderr",
          28    "time": 1.4949557689978974
          29  },{
          30    "data": "[NbConvertApp] Support files will be in __results___files/\n[NbConvertApp] Making directory __results___files\n",
          31    "stream_name": "stderr",
          32    "time": 1.6860545039962744
          33  },{
          34    "data": "[NbConvertApp] Making directory __results___files\n[NbConvertApp] Writing 306796 bytes to __results__.html\n",
          35    "stream_name": "stderr",
          36    "time": 1.6892153889930341
          37  }
          38
          40  Complete. Exited with code 0.
```


Comments (12)

Sort by

All Comments | Hotness

Please sign in to leave a comment.

kudlacz964 • Posted on Latest Version • 2 years ago • Options

2

Why not to use:

```
from sklearn.model_selection import KFold
```

and then:

```
kf = KFold(n_splits=5)

error = []

for train, test in kf.split(traindf):
```

Cristi Vlad • Posted on Latest Version • 2 years ago • Options 1

that's a good way to do it because cross validation will be deprecated in future updates

GauthamS... • Posted on Latest Version • 2 years ago • Options 1

Hi

Thanks for the input. I have continued developing my code locally due to the computational power required for grid search on svm poly kernel. I have changed it. Will upload a new version.

kousalya • Posted on Latest Version • 2 years ago • Options 0

please tell me which version of python we should use
we are getting name*string*to_bool error

rajatarora • Posted on Latest Version • 2 years ago • Options 0

did try to use dnn classifier?

Paulo Lopez • Posted on Latest Version • 2 years ago • Options 0

Great explanation and overall modeling of the cancer predictions.
Have you tried using artificial neural networks?

chayan • Posted on Latest Version • a year ago • Options 0

i am unable to understand everything.....

thisisgurkaran • Posted on Latest Version • 10 months ago • Options 0

Hey great job Buddhini. Learned a lot from this kernel.

thisisgurkaran • Posted on Latest Version • 10 months ago • Options 0

Fit the model:

model.fit(data[predictors],data[outcome])

#Make predictions on training set:
predictions = model.predict(data[predictors])
Can you explain me why you made predictions on the training set and not the test set?? I am new to this and still learning. How can you make predictions on the data that was already fed to the algorithm

Emiliano Nun... • Posted on Latest Version • 9 months ago • Options 0

Great job!, it helped a lot in my first steps, thanks!

Vijay Kumar • Posted on Latest Version • 8 months ago • Options 0

I am using same data and while I typed the following code,
print(data.loc[data.radius*mean==1,['radius*mean', 'texture*mean ','perimeter*mean']])
print(data.loc[(data.radius*mean>=1) | (data.adius*meanmean']])
print(data.loc[(data.radiusmean>=1) & (data.adius*meanmean']])*
print(data.loc[data.radiusmean.isin([5,6,8,10])]) print(data.loc[data.radiusmean.isnull()])
print(data.loc[data.radius_mean.isnull()])
It gave following error:

Empty DataFrame
Columns: [radius*mean, texture*mean , perimeter_mean]
Index: []

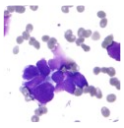
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-loc-reindex-listlike
"""Entry point for launching an IPython kernel.
/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:1367: FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise

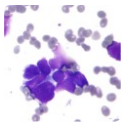
DUrgesh Ku... • Posted on Latest Version • 2 months ago • Options 0

hey can we get those images used in Wisconsin dataset?

Similar Kernels



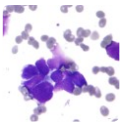
Feature Selection And Data Visualization



Basic Machine Learning With Cancer

ML From Scratch-Part 2

Deep Healthcare Analysis Using BigQuery



Intro To Keras With Breast Cancer Data[ANN]