AbhishekMali21 / LIVER-PATIENT-ANALYSIS Public

Code Issues Pull requests Actions Projects Security Insights

master

LIVER-PATIENT-ANALYSIS / Liver Patient Analysis Notebook.ipynb

AbhishekMali21 Liver Patient Analysis Notebook History

1 contributor

3625 lines (3625 sloc) | 695 KB

👉 👉 👉 👉 LIVER PATIENT ANALYSIS & PREDICTION 👈 👈 👈 👈

1. Data Analysis:

This is in general looking at the data to figure out whats going on. Inspect the data: Check whether there is any missing data, irrelevant data and do a cleanup.

2. Data Visualization:

3. Feature selection.

4. Search for any trends, relations & correlations.

5. Draw an inference and predict whether the patient can be identified to be having liver disease or not

In [1]:

```
#Import all required libraries for reading data, analysing and visualiz

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
```

Data Analysis

In [2]:

```
#Read the training & test data
# liver_df = pd.read_csv('liver_patient.csv')
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage.
# You might want to remove those credentials before you share your note
client_4d3b180310594496b884c11ba3138863 = ibm_boto3.client(service_name
    ibm_api_key_id='dukeFNtwWPtt4DcrXh6HLxDSUIEYxGUMG-xsnXmY5EDW',
    ibm_auth_endpoint="https://iam.eu-gb.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.
```

```
body = client_4d3b180310594496b884c11ba3138863.get_object(Bucket='socia
# add missing __iter__ method, so pandas accepts body as file-like obje
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __i

liver_df= pd.read_csv(body)
liver_df.head()
```

Out[2]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Amin
0	65	Female	0.7	0.1	187	
1	62	Male	10.9	5.5	699	
2	62	Male	7.3	4.1	490	
3	58	Male	1.0	0.4	182	
4	72	Male	3.9	2.0	195	

In [3]:

```
#Top 5 rows of the dataset

liver_df.head()
```

Out[3]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Amin
0	65	Female	0.7	0.1	187	
1	62	Male	10.9	5.5	699	
2	62	Male	7.3	4.1	490	
3	58	Male	1.0	0.4	182	
4	72	Male	3.9	2.0	195	

In [4]:

```
# To get a concise summary of the dataframe

liver_df.info()
```

```
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
Age                           583 non-null int64
Gender                        583 non-null object
Total_Bilirubin               583 non-null float64
Direct_Bilirubin              583 non-null float64
Alkaline_Phosphotase          583 non-null int64
Alamine_Aminotransferase      583 non-null int64
Aspartate_Aminotransferase    583 non-null int64
Total_Protiens                583 non-null float64
Albumin                       583 non-null float64
Albumin_and_Globulin_Ratio    579 non-null float64
Dataset                       583 non-null int64
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

Here are the observations from the dataset:

- Only gender is non-numeric veriable. All others are numeric.
- There are 10 features and 1 output - dataset. Value 1 indicates that the patient has liver disease and 0 indicates the patient does not have liver disease.

In [5]:

```
# Statistical information about NUMERICAL columns in the dataset

liver_df.describe(include='all')
```

Out[5]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Al
count	583.000000	583	583.000000	583.000000	583.000000	
unique	NaN	2	NaN	NaN	NaN	
top	NaN	Male	NaN	NaN	NaN	
freq	NaN	441	NaN	NaN	NaN	
mean	44.746141	NaN	3.298799	1.486106	290.576329	
std	16.189833	NaN	6.209522	2.808498	242.937989	
min	4.000000	NaN	0.400000	0.100000	63.000000	
25%	33.000000	NaN	0.800000	0.200000	175.500000	
50%	45.000000	NaN	1.000000	0.300000	208.000000	
75%	58.000000	NaN	2.600000	1.300000	298.000000	
max	90.000000	NaN	75.000000	19.700000	2110.000000	

- We can see that there are missing values for Albumin_and_Globulin_Ratio as only 579 entries have valid values indicating 4 missing values.
- Gender has only 2 values - Male/Female

In [6]:

```
# Features of the dataset (Labels)

liver_df.columns
```

Out[6]:

```
Index(['Age', 'Gender', 'Total_Bilirubin', 'Direct_Bilirubin',
       'Alkaline_Phosphotase', 'Alamine_Aminotransferase',
       'Aspartate_Aminotransferase', 'Total_Protiens', 'Albumin',
       'Albumin_and_Globulin_Ratio', 'Dataset'],
      dtype='object')
```

In [7]:

```
# Check for any null values

liver_df.isnull().sum()
```

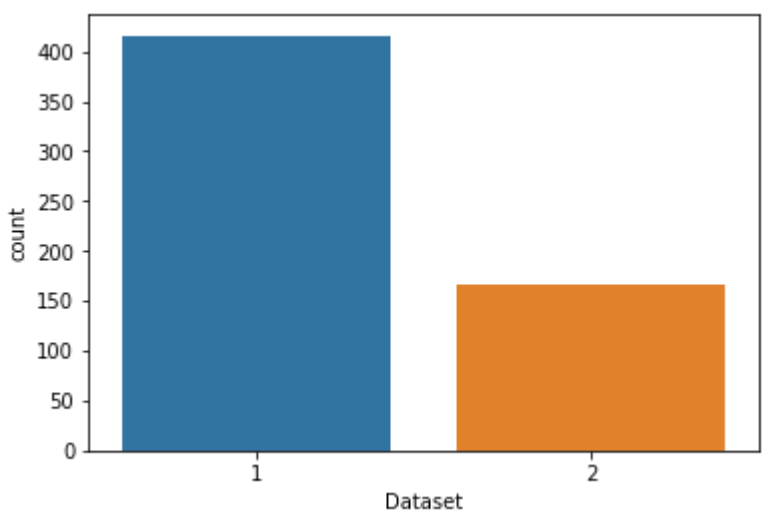
Out[7]:
```
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alamine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    4
Dataset                       0
dtype: int64
```

- The only data that is null is the Albumin_and_Globulin_Ratio - Only 4 rows are null. Lets see whether this is an important feature

Data Visualization

In [8]:
```
# Frequency of patients diagnosed and not diagnoised with liver disease

sns.countplot(data=liver_df, x = 'Dataset', label='Count')

LD, NLD = liver_df['Dataset'].value_counts()
print('Number of patients diagnosed with liver disease: ',LD)
print('Number of patients not diagnosed with liver disease: ',NLD)
```

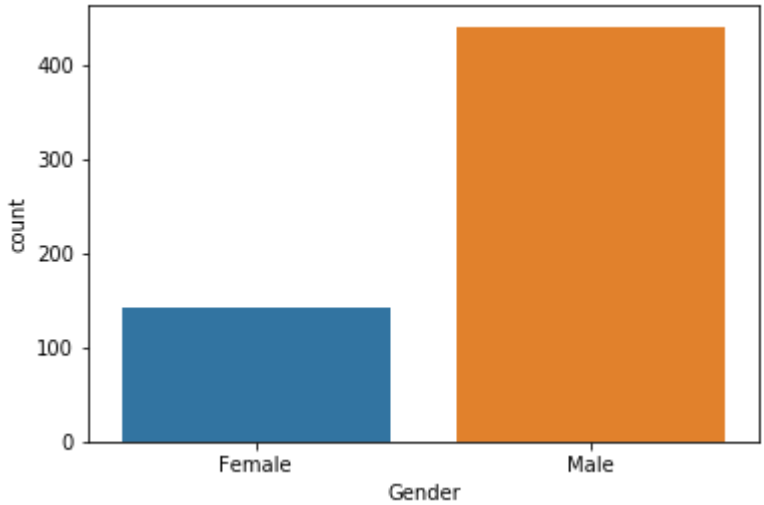
```
Number of patients diagnosed with liver disease:  416
Number of patients not diagnosed with liver disease:  167
/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/seaborn/catego
rical.py:1460: FutureWarning: remove_na is deprecated and is a private
function. Do not use.
  stat_data = remove_na(group_data)
```



In [9]:
```
# Frequency of patients based on their gender

sns.countplot(data=liver_df, x = 'Gender', label='Count')

M, F = liver_df['Gender'].value_counts()
```

```
M, F = liver_df['Gender'].value_counts()
print('Number of patients that are male: ',M)
print('Number of patients that are female: ',F)
```

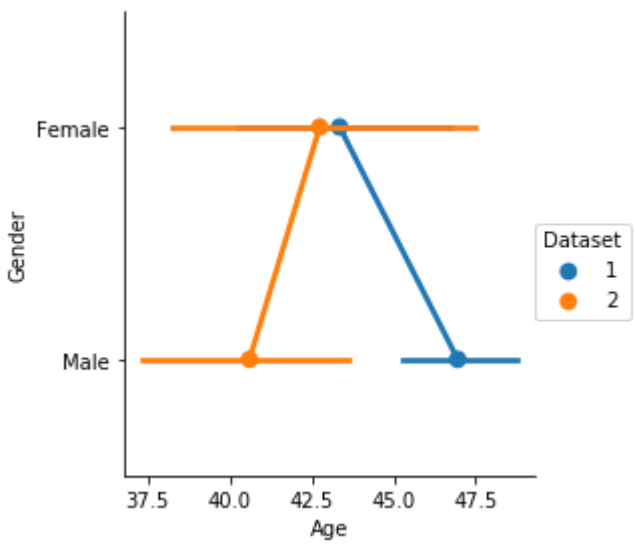
```
/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/seaborn/catego
rical.py:1460: FutureWarning: remove_na is deprecated and is a private
function. Do not use.
  stat_data = remove_na(group_data)
Number of patients that are male:  441
Number of patients that are female:  142
```



In [10]:

```
sns.factorplot(x="Age", y="Gender", hue="Dataset", data=liver_df);
```

```
/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/seaborn/catego
rical.py:1508: FutureWarning: remove_na is deprecated and is a private
function. Do not use.
  stat_data = remove_na(group_data[hue_mask])
```



- Age seems to be a factor for liver disease for both male and female genders

In [11]:

```
liver_df[['Gender', 'Dataset','Age']].groupby(['Dataset','Gender'], as_
```

Out[11]:

	Dataset	Gender	Age
2	2	Female	50
3	2	Male	117
0	1	Female	92
1	1	Male	324

In [12]:

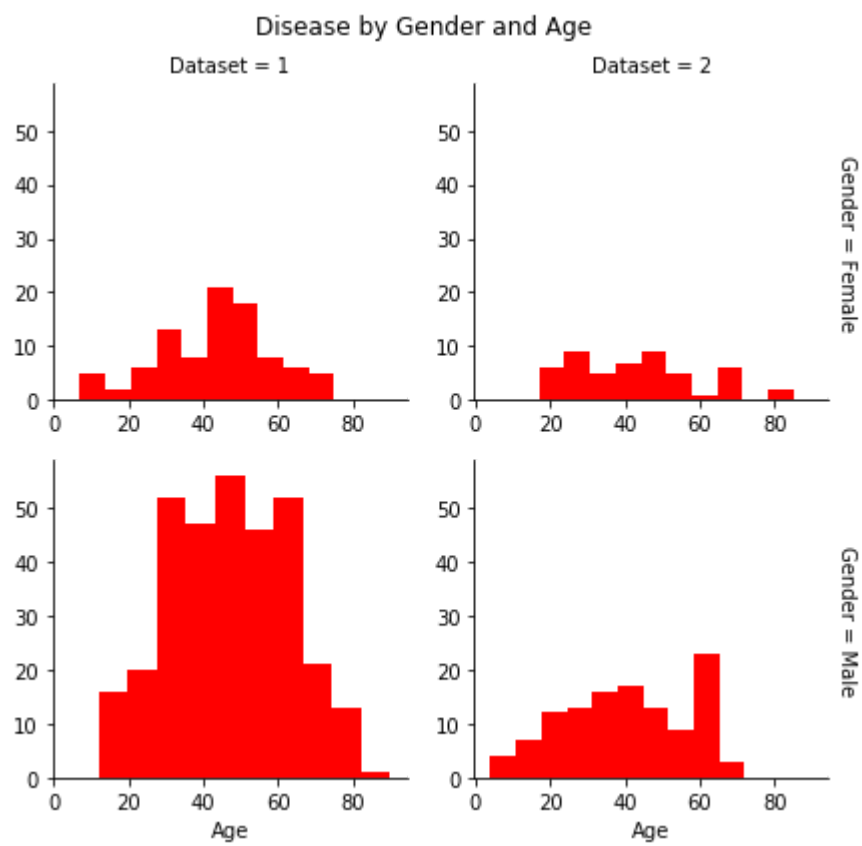
```
liver_df[['Gender', 'Dataset','Age']].groupby(['Dataset','Gender'], as_
```

Out[12]:

	Dataset	Gender	Age
2	2	Female	42.740000
3	2	Male	40.598291
0	1	Female	43.347826
1	1	Male	46.950617

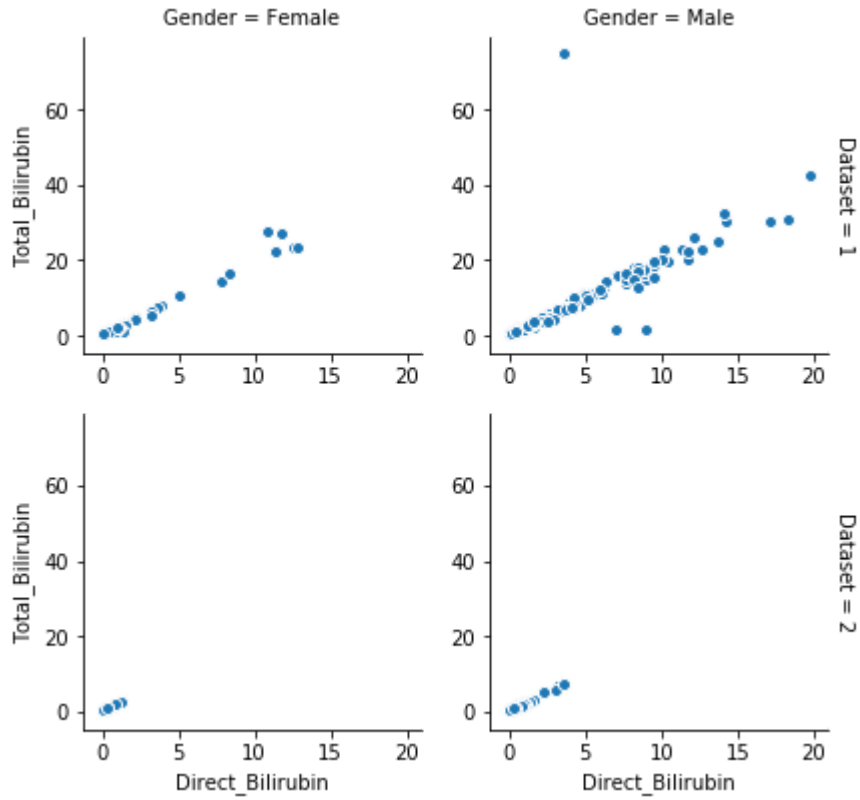
In [13]:

```
g = sns.FacetGrid(liver_df, col="Dataset", row="Gender", margin_titles=
g.map(plt.hist, "Age", color="red")
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Disease by Gender and Age');
```



In [14]:

```
g = sns.FacetGrid(liver_df, col="Gender", row="Dataset", margin_titles=
g.map(plt.scatter,"Direct_Bilirubin", "Total_Bilirubin", edgecolor="w")
plt.subplots_adjust(top=0.9)
```

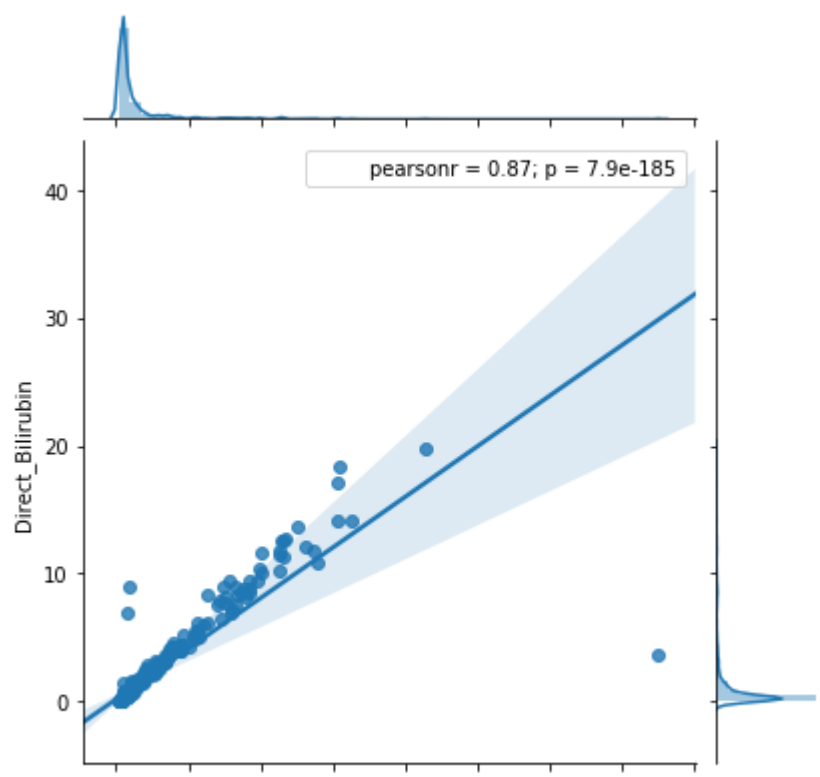


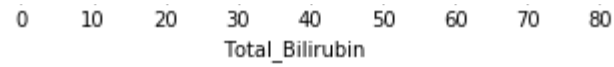
- There seems to be direct relationship between Total_Bilirubin and Direct_Bilirubin. We have the possibility of removing one of this feature.

In [15]:

```
sns.jointplot("Total_Bilirubin", "Direct_Bilirubin", data=liver_df, kin
```

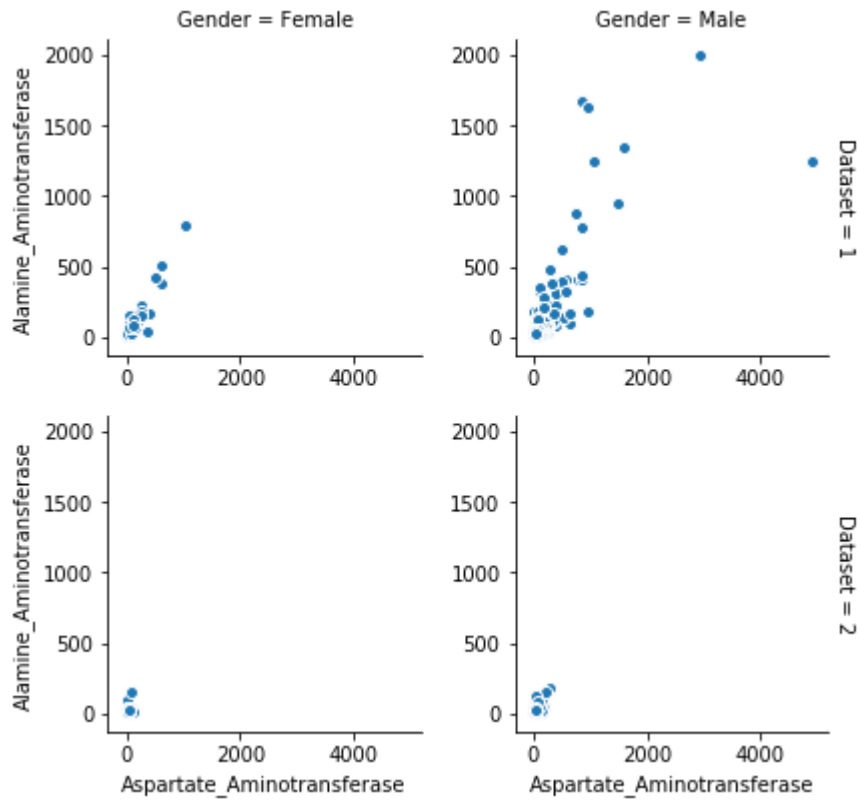
Out[15]:





In [16]:

```
g = sns.FacetGrid(liver_df, col="Gender", row="Dataset", margin_titles=
g.map(plt.scatter,"Aspartate_Aminotransferase", "Alamine_Aminotransfera
plt.subplots_adjust(top=0.9)
```

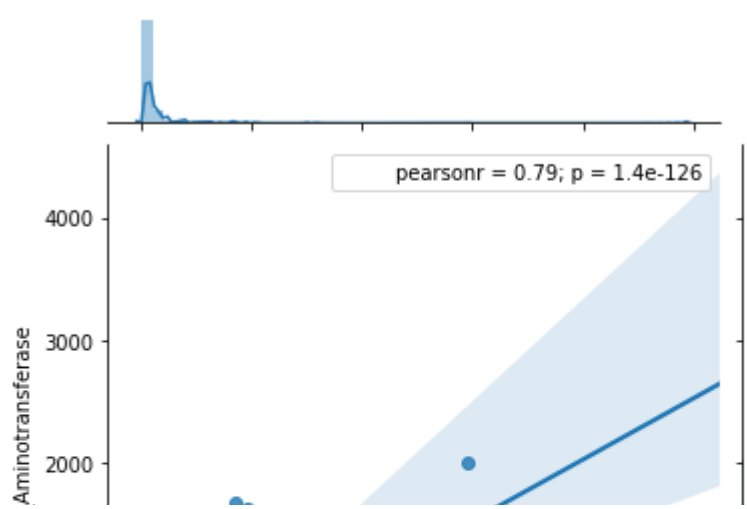


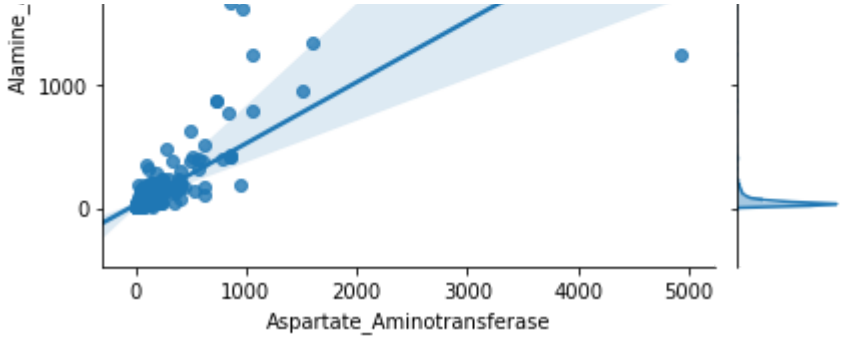
- There is linear relationship between Aspartate_Aminotransferase and Alamine_Aminotransferase and the gender. We have the possibility of removing one of this feature.

In [17]:

```
sns.jointplot("Aspartate_Aminotransferase", "Alamine_Aminotransferase",
```

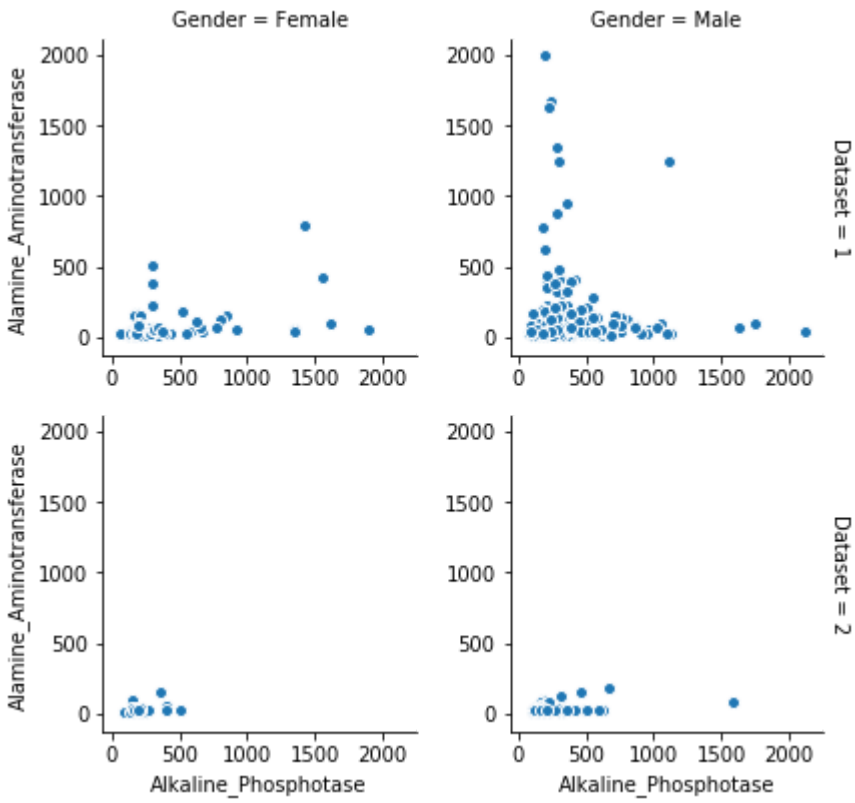
Out[17]:





In [18]:

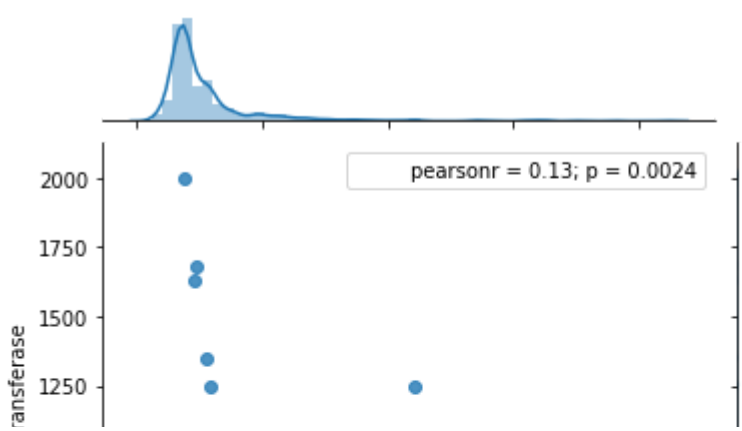
```
g = sns.FacetGrid(liver_df, col="Gender", row="Dataset", margin_titles=
g.map(plt.scatter,"Alkaline_Phosphotase", "Alamine_Aminotransferase",
plt.subplots_adjust(top=0.9)
```

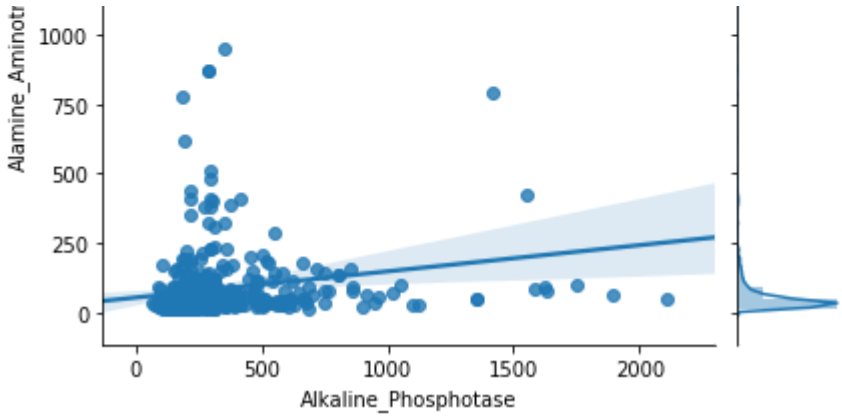


In [19]:

```
sns.jointplot("Alkaline_Phosphotase", "Alamine_Aminotransferase", data=
```

Out[19]:

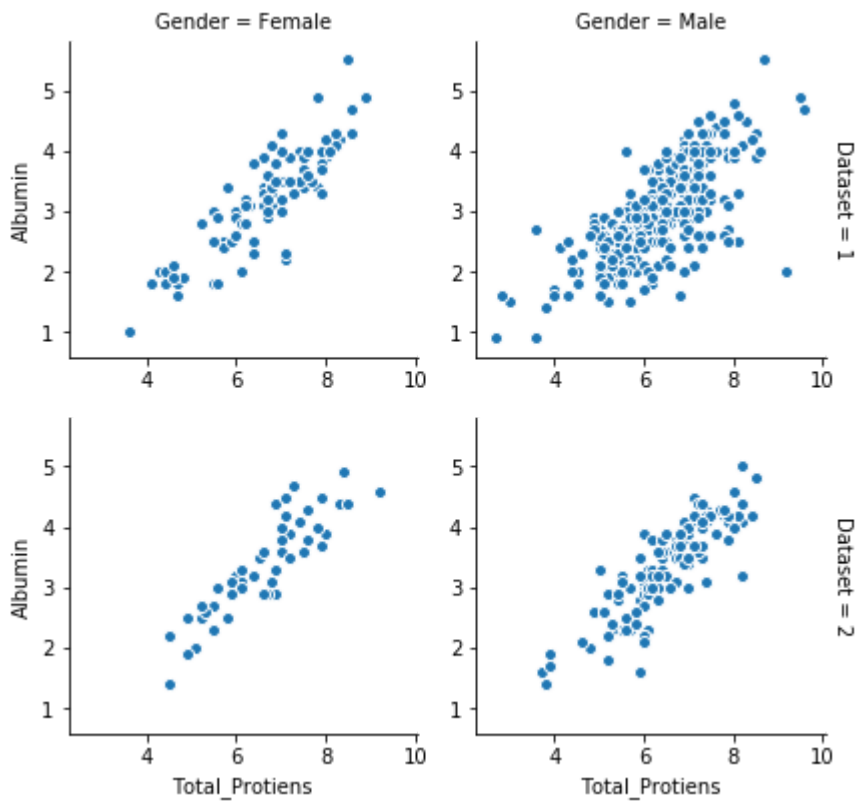




- No linear correlation between Alkaline_Phosphotase and Alamine_Aminotransferase

In [20]:

```
g = sns.FacetGrid(liver_df, col="Gender", row="Dataset", margin_titles=
g.map(plt.scatter,"Total_Protiens", "Albumin",  edgecolor="w")
plt.subplots_adjust(top=0.9)
```

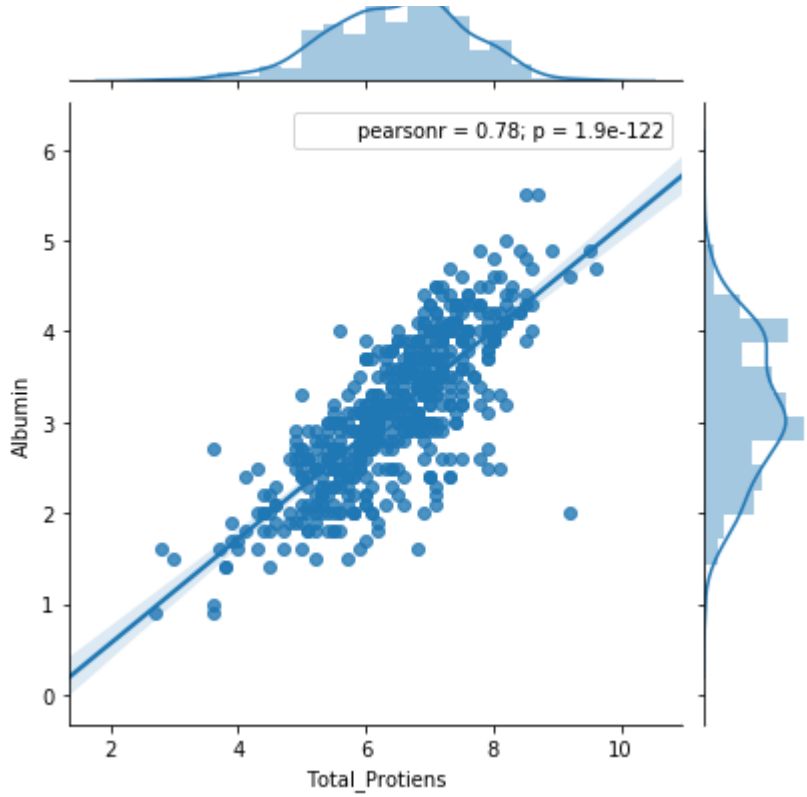


- There is linear relationship between Total_Protiens and Albumin and the gender. We have the possibility of removing one of this feature.

In [21]:

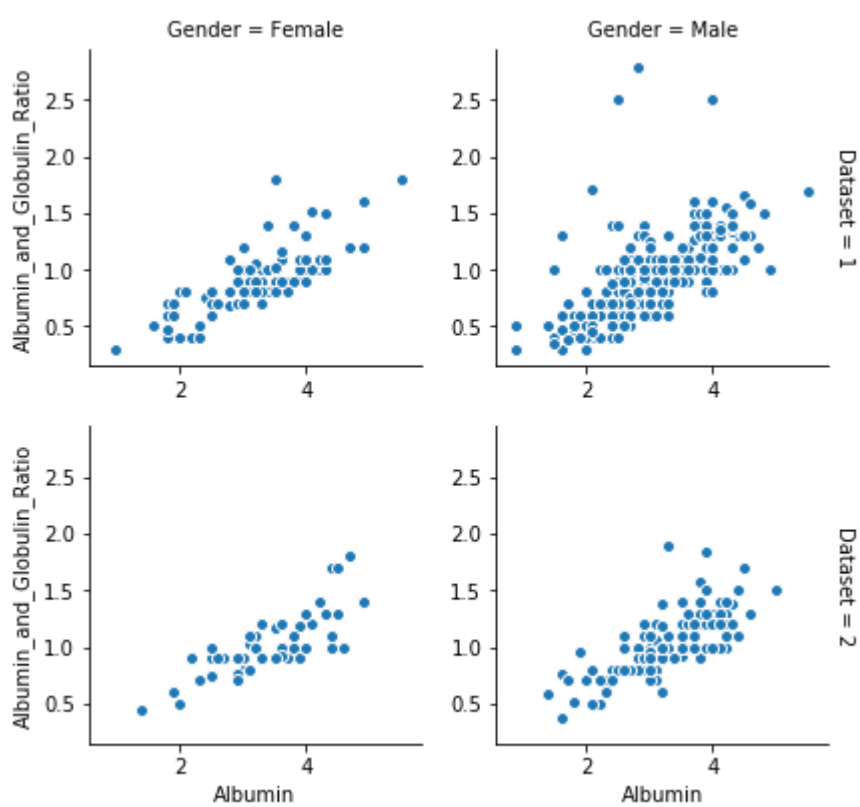
```
sns.jointplot("Total_Protiens", "Albumin", data=liver_df, kind="reg")
```

Out[21]:



In [22]:

```
g = sns.FacetGrid(liver_df, col="Gender", row="Dataset", margin_titles=
g.map(plt.scatter,"Albumin", "Albumin_and_Globulin_Ratio",  edgecolor="
plt.subplots_adjust(top=0.9)
```

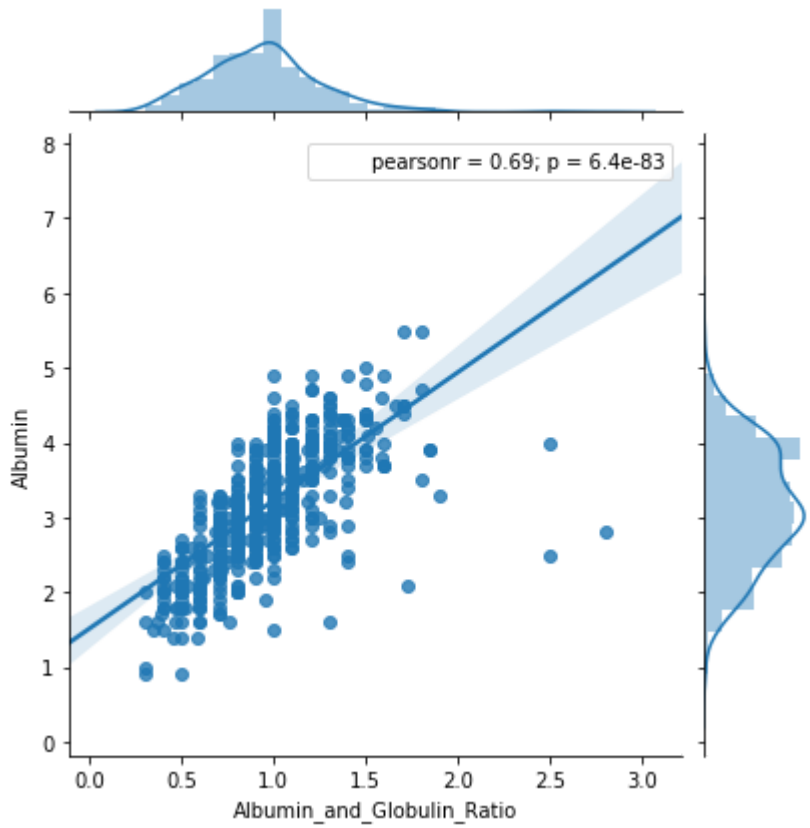


- There is linear relationship between Albumin_and_Globulin_Ratio and Albumin. We have the possibility of removing one of this feature.

In [23]:

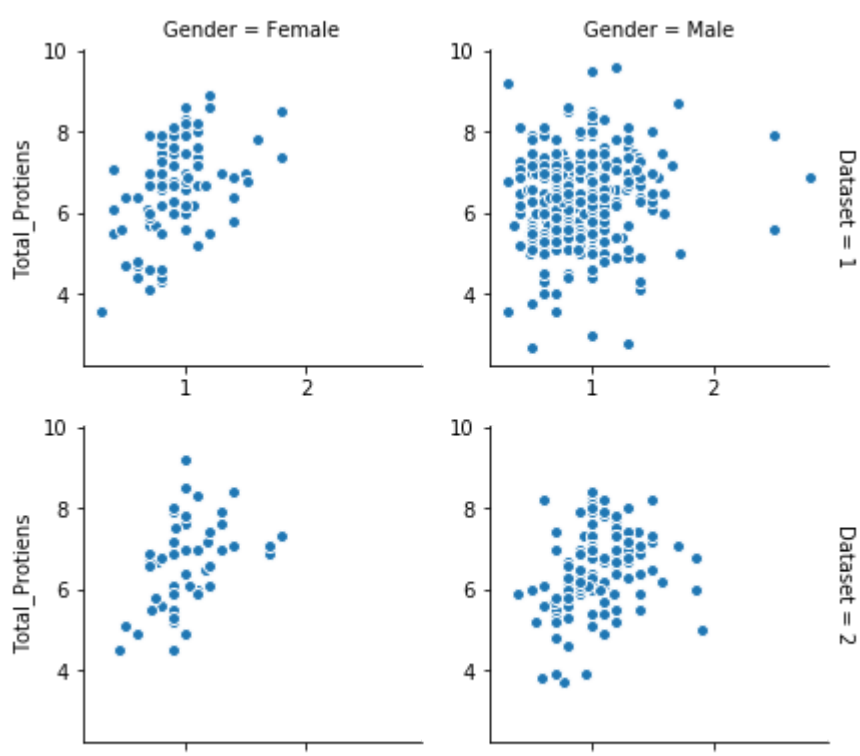
```
sns.jointplot("Albumin_and_Globulin_Ratio", "Albumin", data=liver_df, k
```

Out[23]:



In [24]:

```
g = sns.FacetGrid(liver_df, col="Gender", row="Dataset", margin_titles=
g.map(plt.scatter,"Albumin_and_Globulin_Ratio", "Total_Protiens",  edge
plt.subplots_adjust(top=0.9)
```



1 2 1 2
Albumin_and_Globulin_Ratio Albumin_and_Globulin_Ratio

Observation:

From the above jointplots and scatterplots, we find direct relationship between the following features:

- Direct_Bilirubin & Total_Bilirubin
- Aspartate_Aminotransferase & Alamine_Aminotransferase
- Total_Protiens & Albumin
- Albumin_and_Globulin_Ratio & Albumin

Hence, we can very well find that we can omit one of the features. I'm going to keep the follwing features:

- Total_Bilirubin
- Alamine_Aminotransferase
- Total_Protiens
- Albumin_and_Globulin_Ratio
- Albumin

In [25]:
```
liver_df.head(3)
```

Out[25]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Amin
0	65	Female	0.7	0.1	187	
1	62	Male	10.9	5.5	699	
2	62	Male	7.3	4.1	490	

- Convert categorical variable "Gender" to indicator variables

In [26]:
```
pd.get_dummies(liver_df['Gender'], prefix = 'Gender').head()
```

Out[26]:

	Gender_Female	Gender_Male
0	1	0
1	0	1
2	0	1
3	0	1
4	0	1

In [27]:
```
liver_df = pd.concat([liver_df,pd.get_dummies(liver_df['Gender'], prefi
```

In [28]:
```
liver_df.head()
```

Out[28]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Amin
0	65	Female	0.7	0.1	187	
1	62	Male	10.9	5.5	699	
2	62	Male	7.3	4.1	490	
3	58	Male	1.0	0.4	182	
4	72	Male	3.9	2.0	195	

In [29]:
```
liver_df.describe()
```

Out[29]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Am
count	583.000000	583.000000	583.000000	583.000000	
mean	44.746141	3.298799	1.486106	290.576329	
std	16.189833	6.209522	2.808498	242.937989	
min	4.000000	0.400000	0.100000	63.000000	
25%	33.000000	0.800000	0.200000	175.500000	
50%	45.000000	1.000000	0.300000	208.000000	
75%	58.000000	2.600000	1.300000	298.000000	
max	90.000000	75.000000	19.700000	2110.000000	

- Finding the null values in 'Albumin_and_Globulin_Ratio'

In [30]:
```
liver_df[liver_df['Albumin_and_Globulin_Ratio'].isnull()]
```

Out[30]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_An
209	45	Female	0.9	0.3	189	
241	51	Male	0.8	0.2	230	
253	35	Female	0.6	0.2	180	
312	27	Male	1.3	0.6	106	

In [31]:
```
liver_df["Albumin_and_Globulin_Ratio"] = liver_df.Albumin_and_Globulin_
```

In [32]:

```
#liver_df[liver_df['Albumin_and_Globulin_Ratio'] == 0.9470639032815201]
```

In [33]:

```
# The input variables/features are all the inputs except Dataset.
# The prediction or label is 'Dataset' that determines whether the pati
# Dropping Gender and Dataset

X = liver_df.drop(['Gender','Dataset'], axis=1)
X.head(3)
```

Out[33]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransfer
0	65	0.7	0.1	187	
1	62	10.9	5.5	699	
2	62	7.3	4.1	490	

In [34]:

```
y = liver_df['Dataset']

# 1 for liver disease; 2 for no liver disease
```

In [35]:

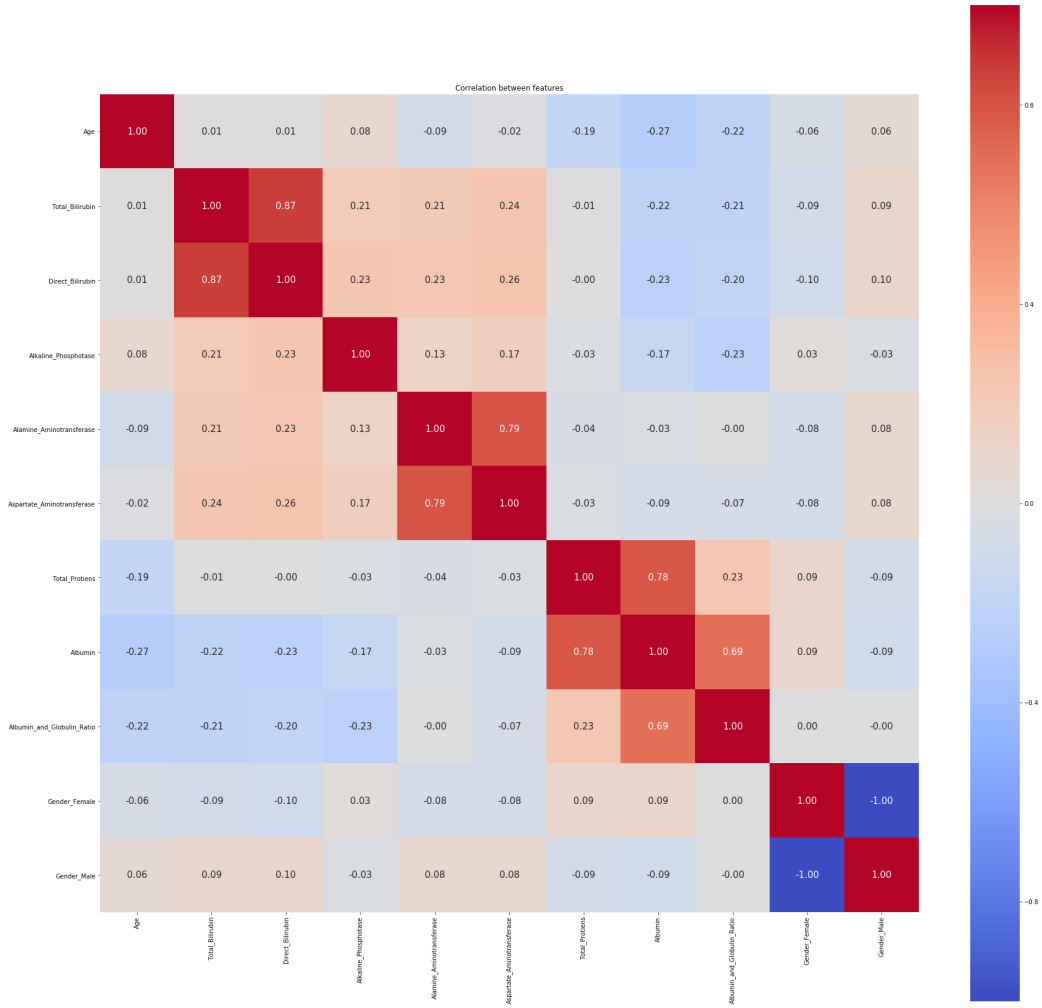
```
# Correlation

liver_corr = X.corr()
liver_corr
```

Out[35]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosp
Age	1.000000	0.011763	0.007529	0.
Total_Bilirubin	0.011763	1.000000	0.874618	0.
Direct_Bilirubin	0.007529	0.874618	1.000000	0.
Alkaline_Phosphotase	0.080425	0.206669	0.234939	1.
Alamine_Aminotransferase	-0.086883	0.214065	0.233894	0.
Aspartate_Aminotransferase	-0.019910	0.237831	0.257544	0.
Total_Protiens	-0.187461	-0.008099	-0.000139	-0.
Albumin	-0.265924	-0.222250	-0.228531	-0.
Albumin_and_Globulin_Ratio	-0.216089	-0.206159	-0.200004	-0.
Gender_Female	-0.056560	-0.089291	-0.100436	0.
Gender_Male	0.056560	0.089291	0.100436	-0.

In [36]:

```
plt.figure(figsize=(30, 30))
sns.heatmap(liver_corr, cbar = True,  square = True, annot=True, fmt= '
plt.title('Correlation between features');
```



The above correlation also indicates the following correlation

- Total_Protiens & Albumin
- Alamine_Aminotransferase & Aspartate_Aminotransferase
- Direct_Bilirubin & Total_Bilirubin
- There is some correlation between Albumin_and_Globulin_Ratio and Albumin. But its not as high as Total_Protiens & Albumin

Machine Learning

In [37]:

```
# Importing modules
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
```

In [38]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
print (X_train.shape)
print (y_train.shape)
print (X_test.shape)
print (y_test.shape)
```

```
(408, 11)
(408,)
(175, 11)
(175,)
```

Logistic Regression

In [39]:

```
# Create logistic regression object

logreg = LogisticRegression()
```

In [40]:

```
# Train the model using the training sets and check score

logreg.fit(X_train, y_train)
```

Out[40]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=
1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.00
01,
          verbose=0, warm_start=False)
```

In [41]:

```
#Predict Output

log_predicted= logreg.predict(X_test)
logreg_score = round(logreg.score(X_train, y_train) * 100, 2)
logreg_score_test = round(logreg.score(X_test, y_test) * 100, 2)
```

In [42]:

```
#Equation coefficient and Intercept

print('Logistic Regression Training Score: \n', logreg_score)
print('Logistic Regression Test Score: \n', logreg_score_test)
print('Coefficient: \n', logreg.coef_)
print('Intercept: \n', logreg.intercept_)
print('Accuracy: \n', accuracy_score(y_test,log_predicted))
print('Confusion Matrix: \n', confusion_matrix(y_test,log_predicted))
print('Classification Report: \n', classification_report(y_test,log_pre
```

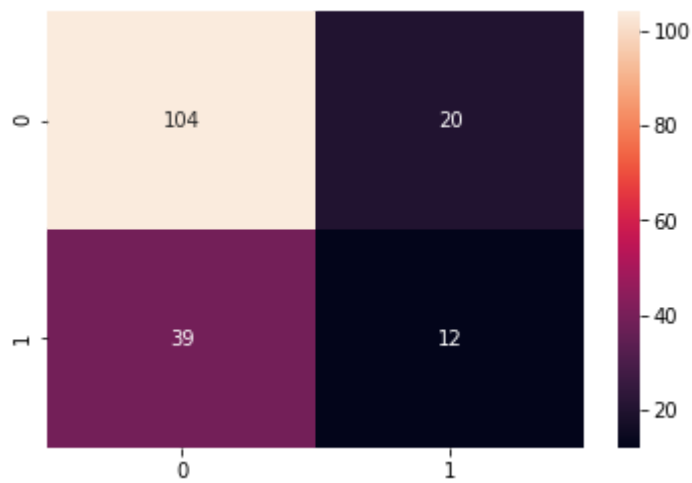
```
Logistic Regression Training Score:
 73.53
Logistic Regression Test Score:
 66.29
Coefficient:
```

```
Coefficient:
 [[-0.01330864 -0.02731202 -0.47915889 -0.00100509 -0.0105125  -0.00311
187
  -0.20786575  0.21451407  0.66623491  0.4855822   0.24378845]]
Intercept:
 [ 0.72937065]
Accuracy:
 0.662857142857
Confusion Matrix:
 [[104  20]
 [ 39  12]]
Classification Report:
              precision    recall  f1-score   support

          1       0.73      0.84      0.78       124
          2       0.38      0.24      0.29        51

avg / total       0.62      0.66      0.64       175
```

In [43]:

```
sns.heatmap(confusion_matrix(y_test,log_predicted),annot=True,fmt="d")
```

Out[43]:



In [44]:

```
coeff_df = pd.DataFrame(X.columns)
coeff_df.columns = ['Feature']
coeff_df["Correlation"] = pd.Series(logreg.coef_[0])
pd.Series(logreg.coef_[0])

coeff_df.sort_values(by='Correlation', ascending=False)
```

Out[44]:

	Feature	Correlation
8	Albumin_and_Globulin_Ratio	0.666235
9	Gender_Female	0.485582
10	Gender_Male	0.243788
7	Albumin	0.214514
3	Alkaline_Phosphotase	-0.001005
5	Aspartate_Aminotransferase	-0.003112

4	Alamine_Aminotransferase	-0.010512
0	Age	-0.013309
1	Total_Bilirubin	-0.027312
6	Total_Protiens	-0.207866
2	Direct_Bilirubin	-0.479159

Gaussian Naive Bayes

In [45]:

```
# Create gaussian object

gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
```

Out[45]: GaussianNB(priors=None)

In [46]:

```
#Predict Output
gauss_predicted = gaussian.predict(X_test)
```

In [47]:

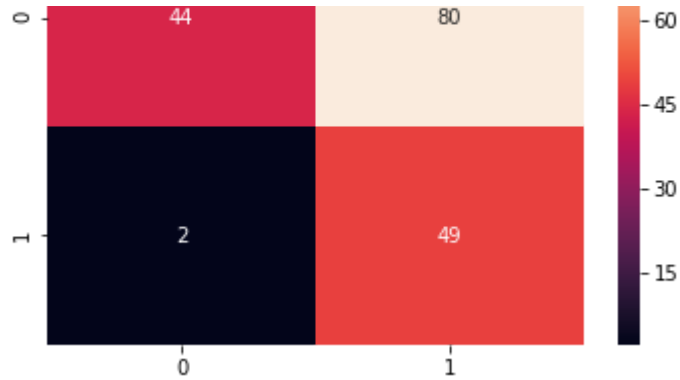
```
gauss_score = round(gaussian.score(X_train, y_train) * 100, 2)
gauss_score_test = round(gaussian.score(X_test, y_test) * 100, 2)
print('Gaussian Score: \n', gauss_score)
print('Gaussian Test Score: \n', gauss_score_test)
print('Accuracy: \n', accuracy_score(y_test, gauss_predicted))
print(confusion_matrix(y_test,gauss_predicted))
print(classification_report(y_test,gauss_predicted))
```

```
Gaussian Score:
 56.13
Gaussian Test Score:
 53.14
Accuracy:
 0.531428571429
[[44 80]
 [ 2 49]]
             precision    recall  f1-score   support

          1       0.96      0.35      0.52       124
          2       0.38      0.96      0.54        51

avg / total       0.79      0.53      0.53       175
```

In [48]:

```
sns.heatmap(confusion_matrix(y_test,gauss_predicted),annot=True,fmt="d"
```

Out[48]:



Random Forest

In [49]:

```
# create random_forest object

random_forest = RandomForestClassifier(max_depth=3,n_estimators=56,crit
random_forest.fit(X_train, y_train)
```

Out[49]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='en
tropy',
            max_depth=3, 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=56, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [50]:

```
#Predict Output

rf_predicted = random_forest.predict(X_test)
```

In [51]:

```
random_forest_score = round(random_forest.score(X_train, y_train) * 100
random_forest_score_test = round(random_forest.score(X_test, y_test) *
print('Random Forest Score: \n', random_forest_score)
print('Random Forest Test Score: \n', random_forest_score_test)
print('Accuracy: \n', accuracy_score(y_test,rf_predicted))
print(confusion_matrix(y_test,rf_predicted))
print(classification_report(y_test,rf_predicted))
```

```
Random Forest Score:
 75.49
Random Forest Test Score:
 68.57
Accuracy:
 0.685714285714
[[116   8]
 [ 47   4]]
             precision    recall  f1-score   support

          1       0.71      0.94      0.81       124
          2       0.33      0.08      0.13        51

avg / total       0.60      0.69      0.61       175
```

In [52]:
```
finX = liver_df[['Total_Protiens','Albumin', 'Gender_Male']]
finX.head(4)
```

Out[52]:

	Total_Protiens	Albumin	Gender_Male
0	6.8	3.3	0
1	7.5	3.2	1
2	7.0	3.3	1
3	6.8	3.4	1

Logistic Regression

In [53]:
```
X_train, X_test, y_train, y_test = train_test_split(finX, y, test_size=
```

In [54]:
```
# Create logistic regression object

logreg = LogisticRegression()
```

In [55]:
```
# Train the model using the training sets and check score

logreg.fit(X_train, y_train)
```

Out[55]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=
1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.00
01,
          verbose=0, warm_start=False)
```

In [56]:
```
# Predict Output

log_predicted= logreg.predict(X_test)
```

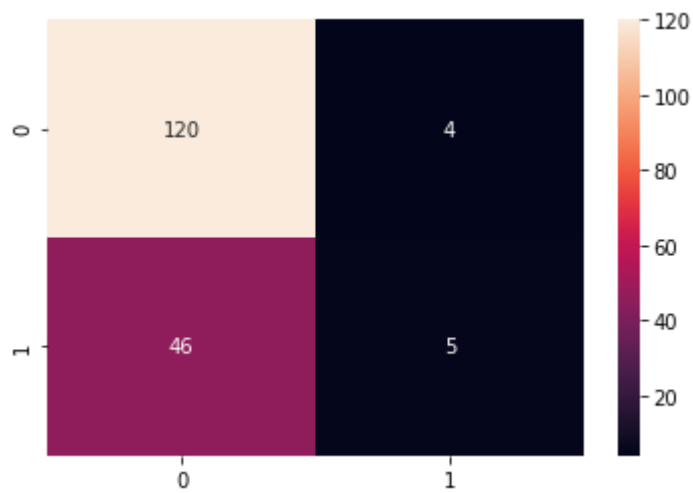
In [57]:
```
logreg_score = round(logreg.score(X_train, y_train) * 100, 2)
logreg_score_test = round(logreg.score(X_test, y_test) * 100, 2)

# Equation coefficient and Intercept

print('Logistic Regression Training Score: \n', logreg_score)
print('Logistic Regression Test Score: \n', logreg_score_test)
print('Coefficient: \n', logreg.coef_)
print('Intercept: \n', logreg.intercept_)
print('Accuracy: \n', accuracy_score(y_test,log_predicted))
print('Confusion Matrix: \n', confusion_matrix(y_test,log_predicted))
print('Classification Report: \n', classification_report(y_test,log_pre
```

```
Logistic Regression Training Score:
 71.08
Logistic Regression Test Score:
 71.43
Coefficient:
 [[-0.59570175  1.09488855 -0.54783114]]
Intercept:
 [-0.1301474]
Accuracy:
 0.714285714286
Confusion Matrix:
 [[120   4]
 [ 46   5]]
Classification Report:
              precision    recall  f1-score   support

          1       0.72      0.97      0.83       124
          2       0.56      0.10      0.17        51

avg / total       0.67      0.71      0.63       175
```

In [58]:

```
sns.heatmap(confusion_matrix(y_test,log_predicted),annot=True,fmt="d")
```

Out[58]:



Decision Tree Classifier

In [59]:

```
# Create decision tree object

dt=DecisionTreeClassifier()
```

In [60]:

```
# Train the model using the training sets and check score

dt.fit(X_train,y_train)
```

Out[60]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=N
one,
```

```
                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, presort=False, random_state=N
one,
                splitter='best')
```

In [61]:

```
# Predict Output

y_pred=dt.predict(X_test)

dt_score = round(dt.score(X_train, y_train) * 100, 2)
dt_test = round(dt.score(X_test, y_test) * 100, 2)
```

In [62]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[62]:

```
0.62857142857142856
```

In [63]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[63]:

```
array([[96, 28],
       [37, 14]])
```

Model evaluation

In [64]:

```
# We can now rank our evaluation of all the models to choose the best o

models = pd.DataFrame({
    'Model': [ 'Logistic Regression', 'Gaussian Naive Bayes','Random Fo
    'Score': [ logreg_score, gauss_score, random_forest_score,dt_score]
    'Test Score': [ logreg_score_test, gauss_score_test, random_forest_
models.sort_values(by='Test Score', ascending=False)
```

Out[64]:

	Model	Score	Test Score
0	Logistic Regression	71.08	71.43
2	Random Forest	75.49	68.57
3	Decision Tree	93.38	62.86
1	Gaussian Naive Bayes	56.13	53.14

In [65]:

```
from watson_machine_learning_client import WatsonMachineLearningAPIClie

wml_credentials = {
   "instance_id": "1c8b2a30-08fb-4355-a71b-ea152c85f5b7",
  "password": "c88f97ec-f410-425e-bd14-114b067ffec2",
  "url": "https://eu-gb.ml.cloud.ibm.com",
  "username": "4478a4bf-e7d3-4311-8e99-b93fc1ace7a0"
}
```

```
client = WatsonMachineLearningAPIClient(wml_credentials)
model_props = {client.repository.ModelMetaNames.AUTHOR_NAME: "Abhishek"
               client.repository.ModelMetaNames.AUTHOR_EMAIL: "abhishek
               client.repository.ModelMetaNames.NAME: "LiverPatient Mod
              }


model=client.repository.store_model(logreg, meta_props=model_props)

published_model_uid = client.repository.get_model_uid(model)
published_model_uid

deployment = client.deployments.create(published_model_uid, name="Patie

scoring_endpoint = client.deployments.get_scoring_url(deployment)
scoring_endpoint
```

```
/opt/conda/envs/DSX-Python35/lib/python3.5/site-packages/sklearn/cross_
validation.py:41: DeprecationWarning: This module was deprecated in ver
sion 0.18 in favor of the model_selection module into which all the ref
actored classes and functions are moved. Also note that the interface o
f the new CV iterators are different from that of this module. This mod
ule will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
2019-07-20 13:19:13,544 - watson_machine_learning_client.metanames - WA
RNING - 'AUTHOR_EMAIL' meta prop is deprecated. It will be ignored.

#######################################################################
################

Synchronous deployment creation for uid: 'dbe6af6a-34d4-41e3-9ef2-d0ed1
5c18aed' started

#######################################################################
################


INITIALIZING
DEPLOY_SUCCESS


-----------------------------------------------------------------------
-------------------------
Successfully finished deployment creation, deployment_uid='d3179405-5d5
a-4b9c-ae50-18b8b14d4c7e'
-----------------------------------------------------------------------
-------------------------
```