# MANOJ KUMAR - 2048015

In [1]:

```
#Importing libraries
import numpy as np
import pandas as pd
```

In [2]:

```
#Importing the visualisation libraries
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:

```
#Reading the data
kidneyDisease = pd.read_csv('kidney_disease.csv')
```

## Exploratory Data Analysis

In [4]:

```
#Shape of the dataset
kidneyDisease.shape
```

Out[4]:

```
(400, 26)
```

In [5]:

```
kidneyDisease.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              400 non-null    int64  
 1   age             391 non-null    float64
 2   bp              388 non-null    float64
 3   sg              353 non-null    float64
 4   al              354 non-null    float64
 5   su              351 non-null    float64
 6   rbc             248 non-null    object 
 7   pc              335 non-null    object 
 8   pcc             396 non-null    object 
 9   ba              396 non-null    object 
 10  bgr             356 non-null    float64
 11  bu              381 non-null    float64
 12  sc              383 non-null    float64
 13  sod             313 non-null    float64
 14  pot             312 non-null    float64
 15  hemo            348 non-null    float64
 16  pcv             330 non-null    object 
 17  wc              295 non-null    object 
 18  rc              270 non-null    object 
 19  htn             398 non-null    object 
 20  dm              398 non-null    object 
 21  cad             398 non-null    object 
 22  appet           399 non-null    object 
 23  pe              399 non-null    object 
 24  ane             399 non-null    object 
 25  classification  400 non-null    object 
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB
```

In [6]:

```
kidneyDisease.describe()
```

Out[6]:

| | id | age | bp | sg | al | su | bgr | |
|---|---|---|---|---|---|---|---|---|
| **count** | 400.000000 | 391.000000 | 388.000000 | 353.000000 | 354.000000 | 351.000000 | 356.000000 | 381 |
| **mean** | 199.500000 | 51.483376 | 76.469072 | 1.017408 | 1.016949 | 0.450142 | 148.036517 | 57 |
| **std** | 115.614301 | 17.169714 | 13.683637 | 0.005717 | 1.352679 | 1.099191 | 79.281714 | 5 |
| **min** | 0.000000 | 2.000000 | 50.000000 | 1.005000 | 0.000000 | 0.000000 | 22.000000 | 1 |
| **25%** | 99.750000 | 42.000000 | 70.000000 | 1.010000 | 0.000000 | 0.000000 | 99.000000 | 27 |
| **50%** | 199.500000 | 55.000000 | 80.000000 | 1.020000 | 0.000000 | 0.000000 | 121.000000 | 42 |
| **75%** | 299.250000 | 64.500000 | 80.000000 | 1.020000 | 2.000000 | 0.000000 | 163.000000 | 6 |
| **max** | 399.000000 | 90.000000 | 180.000000 | 1.025000 | 5.000000 | 5.000000 | 490.000000 | 391 |

## Data pre-processing

In [7]:

```
cols_names={"bp":"blood_pressure",
          "sg":"specific_gravity",
          "al":"albumin",
          "su":"sugar",
          "rbc":"red_blood_cells",
          "pc":"pus_cell",
          "pcc":"pus_cell_clumps",
          "ba":"bacteria",
          "bgr":"blood_glucose_random",
          "bu":"blood_urea",
          "sc":"serum_creatinine",
          "sod":"sodium",
          "pot":"potassium",
          "hemo":"haemoglobin",
          "pcv":"packed_cell_volume",
          "wc":"white_blood_cell_count",
          "rc":"red_blood_cell_count",
          "htn":"hypertension",
          "dm":"diabetes_mellitus",
          "cad":"coronary_artery_disease",
          "appet":"appetite",
          "pe":"pedal_edema",
          "ane":"anemia"}

kidneyDisease.rename(columns=cols_names, inplace=True)
```

In [8]:

```
# Dropping the id column and customization

kidneyDisease['red_blood_cell_count'] = pd.to_numeric(kidneyDisease['red_blood_cell_
                                                      errors='coerce')
kidneyDisease['packed_cell_volume'] = pd.to_numeric(kidneyDisease['packed_cell_volum
                                                    errors='coerce')
kidneyDisease['white_blood_cell_count'] = pd.to_numeric(kidneyDisease['white_blood_c
                                                        errors='coerce')

kidneyDisease.drop(["id"],axis=1,inplace=True)
```

In [9]:

```
kidneyDisease.isnull().sum().sort_values(ascending=False)
```

Out[9]:

```
red_blood_cells            152
red_blood_cell_count       131
white_blood_cell_count     106
potassium                   88
sodium                      87
packed_cell_volume          71
pus_cell                    65
haemoglobin                 52
sugar                       49
specific_gravity            47
albumin                     46
blood_glucose_random        44
blood_urea                  19
serum_creatinine            17
blood_pressure              12
age                          9
bacteria                     4
pus_cell_clumps              4
hypertension                 2
diabetes_mellitus            2
coronary_artery_disease      2
anemia                       1
appetite                     1
pedal_edema                  1
classification               0
dtype: int64
```

In [10]:

```
# Numerical & Categorical features:
```

In [11]:

```
numerical_features = []
categorical_features = []

for i in kidneyDisease.columns:
    if kidneyDisease[i].nunique()>7:
        numerical_features.append(i)
    else:
        categorical_features.append(i)

# Numerical
print(numerical_features)
print("\n")
# Categorical
print(categorical_features)
```

```
['age', 'blood_pressure', 'blood_glucose_random', 'blood_urea', 'serum
_creatinine', 'sodium', 'potassium', 'haemoglobin', 'packed_cell_volum
e', 'white_blood_cell_count', 'red_blood_cell_count']


['specific_gravity', 'albumin', 'sugar', 'red_blood_cells', 'pus_cel
l', 'pus_cell_clumps', 'bacteria', 'hypertension', 'diabetes_mellitu
s', 'coronary_artery_disease', 'appetite', 'pedal_edema', 'anemia', 'c
lassification']
```

In [12]:

```
# Replace incorrect values

kidneyDisease['diabetes_mellitus'] = kidneyDisease['diabetes_mellitus'].replace(to_r


kidneyDisease['coronary_artery_disease'] = kidneyDisease['coronary_artery_disease'].

kidneyDisease['classification'] = kidneyDisease['classification'].replace(to_replace
                                                                          value = 'c
```

Actual work

In [13]:

```
# Duplicating df
df = kidneyDisease

# Filter the Numerical data
df1 = df.filter(numerical_features, axis=1)
```

In [14]:

```
# Case 1 df with NOT NULL Original data
df_notnull = df.dropna().filter(numerical_features, axis=1)
df_notnull
```

Out[14]:

| | age | blood_pressure | blood_glucose_random | blood_urea | serum_creatinine | sodium | potass |
|---|---|---|---|---|---|---|---|
| **3** | 48.0 | 70.0 | 117.0 | 56.0 | 3.8 | 111.0 | |
| **9** | 53.0 | 90.0 | 70.0 | 107.0 | 7.2 | 114.0 | |
| **11** | 63.0 | 70.0 | 380.0 | 60.0 | 2.7 | 131.0 | |
| **14** | 68.0 | 80.0 | 157.0 | 90.0 | 4.1 | 130.0 | |
| **20** | 61.0 | 80.0 | 173.0 | 148.0 | 3.9 | 135.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **395** | 55.0 | 80.0 | 140.0 | 49.0 | 0.5 | 150.0 | |
| **396** | 42.0 | 70.0 | 75.0 | 31.0 | 1.2 | 141.0 | |
| **397** | 12.0 | 80.0 | 100.0 | 26.0 | 0.6 | 137.0 | |
| **398** | 17.0 | 60.0 | 114.0 | 50.0 | 1.0 | 135.0 | |
| **399** | 58.0 | 80.0 | 131.0 | 18.0 | 1.1 | 141.0 | |

158 rows × 11 columns

In [15]:

```
# Case 2 df with Imputed mean data

from sklearn.impute import SimpleImputer
miss_mean_imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
miss_mean_imputer = miss_mean_imputer.fit(df1)
df_imputed = miss_mean_imputer.transform(df1.values)
df_imputed = pd.DataFrame(df_imputed, columns = numerical_features)
df_imputed
```

Out[15]:

| | age | blood_pressure | blood_glucose_random | blood_urea | serum_creatinine | sodium | pot |
|---|---|---|---|---|---|---|---|
| **0** | 48.0 | 80.0 | 121.000000 | 36.0 | 1.2 | 137.528754 | 4 |
| **1** | 7.0 | 50.0 | 148.036517 | 18.0 | 0.8 | 137.528754 | 4 |
| **2** | 62.0 | 80.0 | 423.000000 | 53.0 | 1.8 | 137.528754 | 4 |
| **3** | 48.0 | 70.0 | 117.000000 | 56.0 | 3.8 | 111.000000 | 2 |
| **4** | 51.0 | 80.0 | 106.000000 | 26.0 | 1.4 | 137.528754 | 4 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **395** | 55.0 | 80.0 | 140.000000 | 49.0 | 0.5 | 150.000000 | 4 |
| **396** | 42.0 | 70.0 | 75.000000 | 31.0 | 1.2 | 141.000000 | 3 |
| **397** | 12.0 | 80.0 | 100.000000 | 26.0 | 0.6 | 137.000000 | 4 |
| **398** | 17.0 | 60.0 | 114.000000 | 50.0 | 1.0 | 135.000000 | 4 |
| **399** | 58.0 | 80.0 | 131.000000 | 18.0 | 1.1 | 141.000000 | 3 |

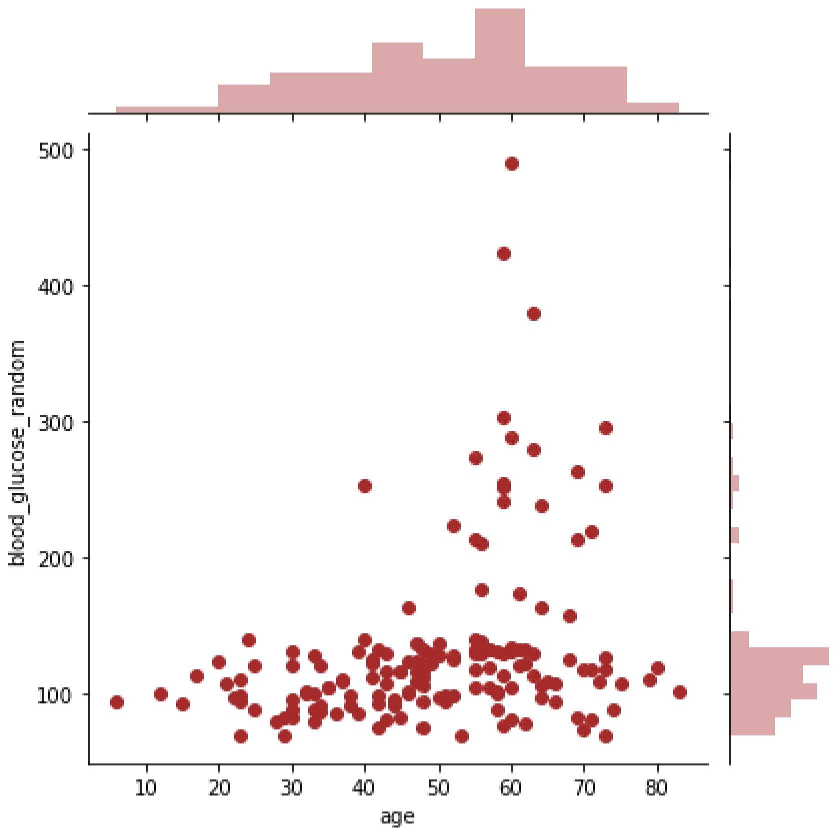
400 rows × 11 columns

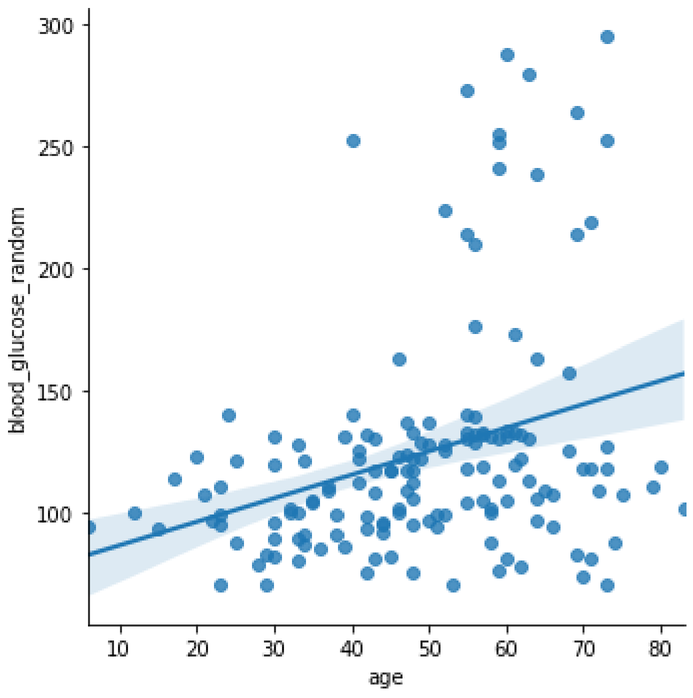
In [16]:

```
#Jointplot
sns.jointplot(x='age',y='blood_glucose_random',data=df_notnull, color='brown')
df_notnull = df_notnull[df_notnull['blood_glucose_random'].between(0, 300)]
sns.lmplot(x='age',y='blood_glucose_random',data=df_notnull)
```

Out[16]:

```
<seaborn.axisgrid.FacetGrid at 0x7feae11b3550>
```





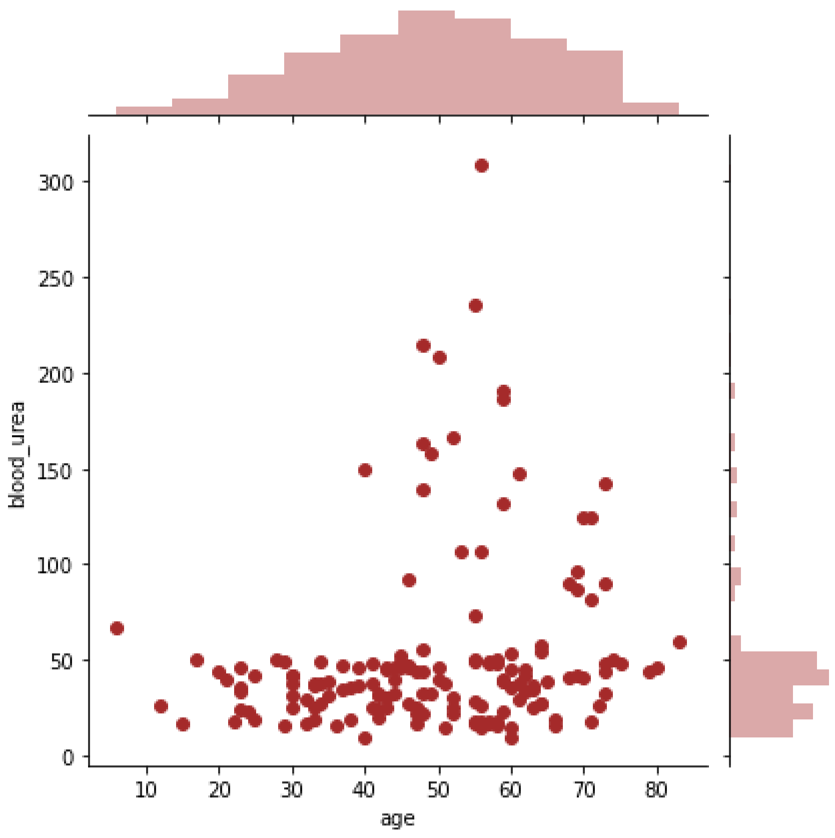
blood_glucose_random
300
250
200
150
100
10
20
30
40
50
60
70
80
age

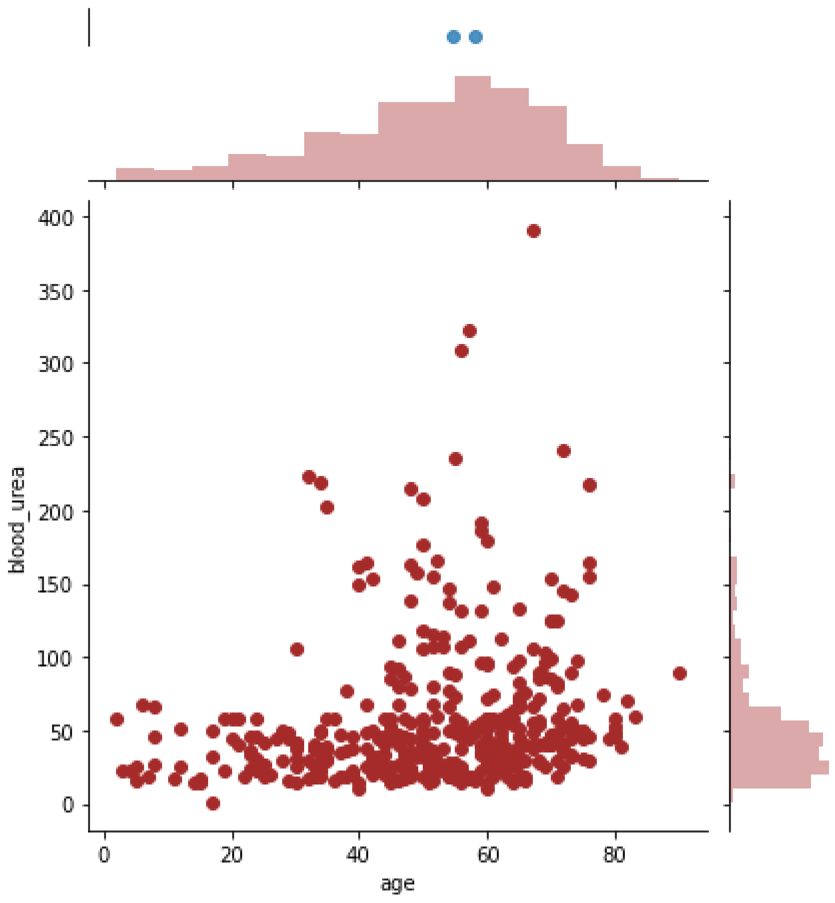
In [17]:

```
#Jointplot
sns.jointplot(x='age',y='blood_urea',data=df_notnull,color='brown')
df_notnull = df_notnull[df_notnull['blood_urea'].between(0, 120)]
sns.lmplot(x='age',y='blood_urea',data=df_notnull)


sns.jointplot(x='age',y='blood_urea',data=df_imputed,color='brown')
df_imputed = df_imputed[df_imputed['blood_urea'].between(0, 120)]
sns.lmplot(x='age',y='blood_urea',data=df_imputed)
# blood_glucose_random blood_urea serum_creatinine sodium
```

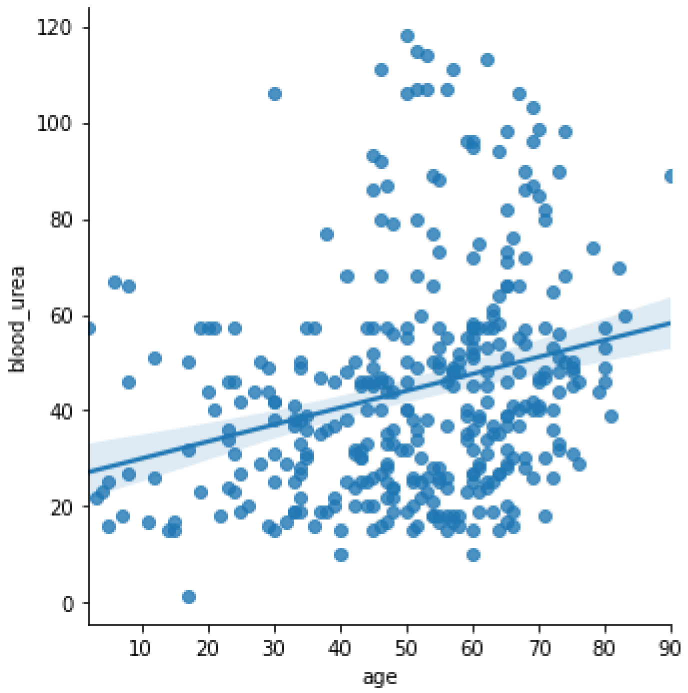
Out[17]:

```
<seaborn.axisgrid.FacetGrid at 0x7feae1979130>
```





400
350
300
250
200
150
100
50
0
blood_urea
0
20
40
60
80
age



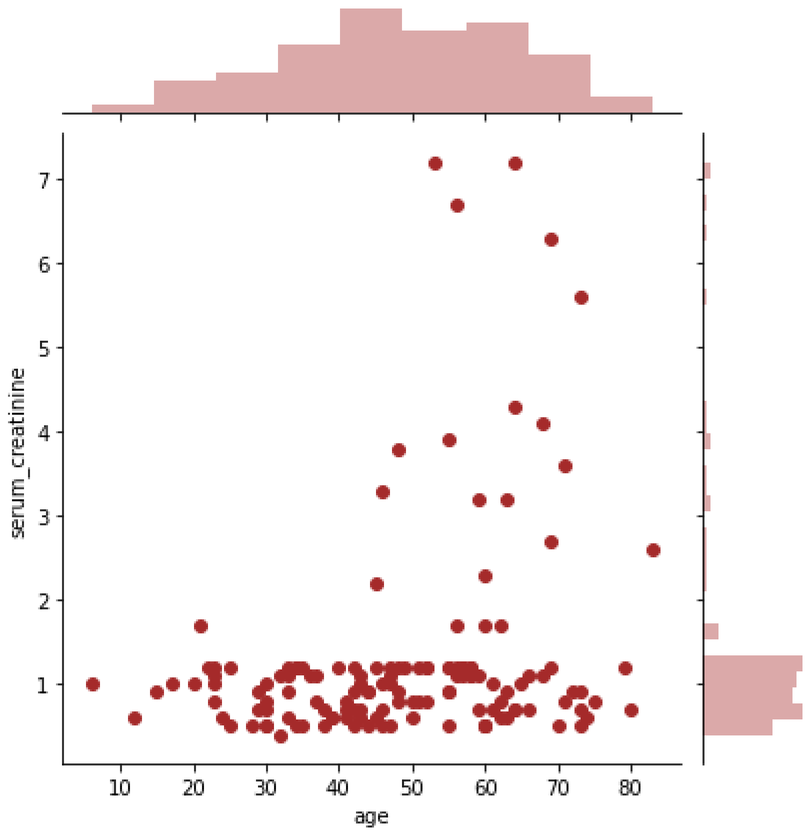
120
100
80
60
40
20
0
blood_urea
10
20
30
40
50
60
70
80
90
age

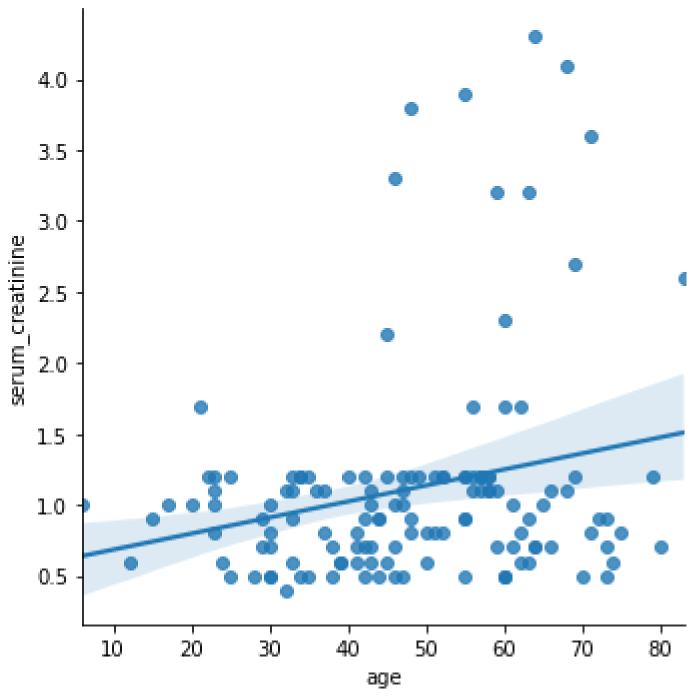
In [18]:

```
#Jointplot
sns.jointplot(x='age',y='serum_creatinine',data=df_notnull,color='brown')
df_notnull = df_notnull[df_notnull['serum_creatinine'].between(0, 5)]
sns.lmplot(x='age',y='serum_creatinine',data=df_notnull)


sns.jointplot(x='age',y='serum_creatinine',data=df_imputed,color='brown')
df_imputed = df_imputed[df_imputed['serum_creatinine'].between(0, 5)]
sns.lmplot(x='age',y='serum_creatinine',data=df_imputed)


# blood_glucose_random blood_urea serum_creatinine sodium
```

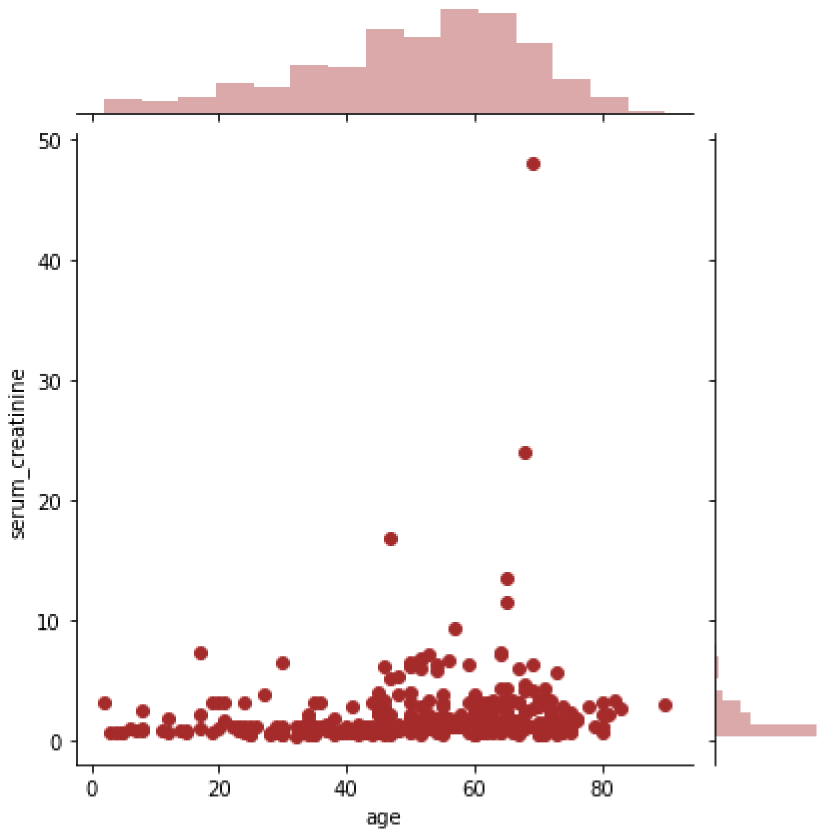
Out[18]:

```
<seaborn.axisgrid.FacetGrid at 0x7feae17497c0>
```

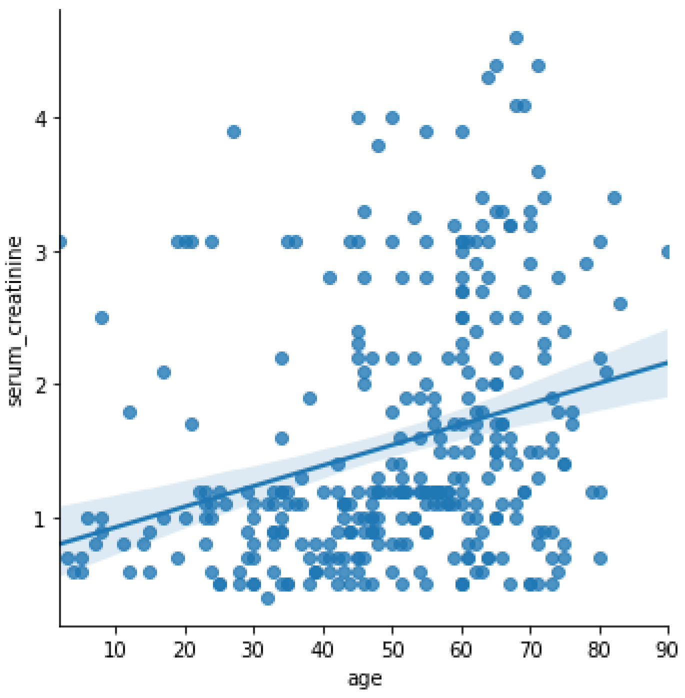




serum_creatinine
age



serum_creatinine
age

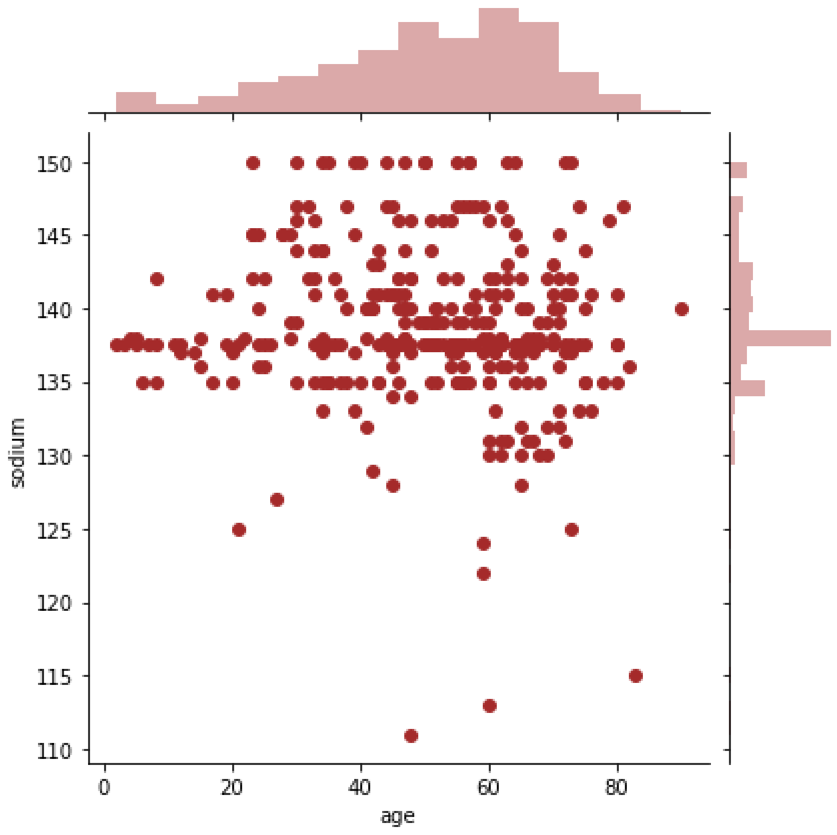

serum_creatinine
age
1
2
3
4
10
20
30
40
50
60
70
80
90

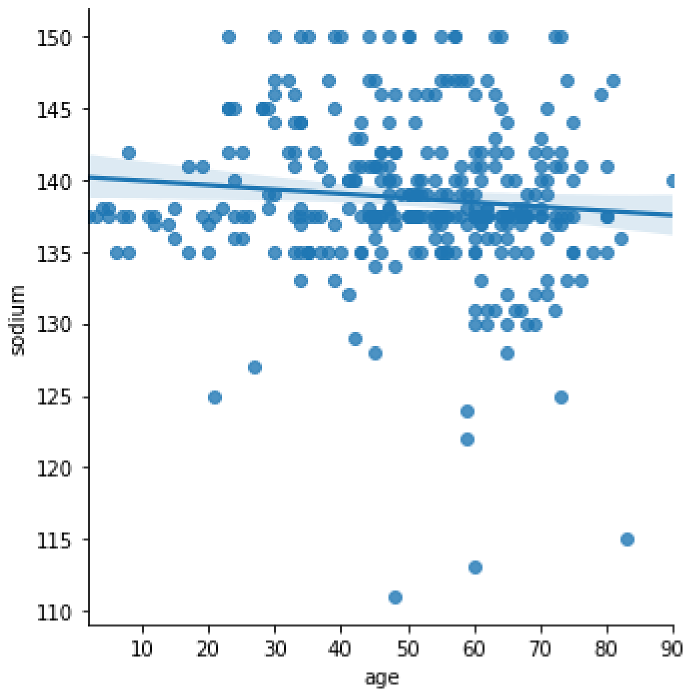
```
In [19]:
```

```
#Jointplot

sns.jointplot(x='age',y='sodium',data=df_imputed,color='brown')
df_imputed = df_imputed[df_imputed['sodium'].between(80, 180)]
sns.lmplot(x='age',y='sodium',data=df_imputed)

# blood_glucose_random blood_urea serum_creatinine sodium
```

```
Out[19]:
```

```
<seaborn.axisgrid.FacetGrid at 0x7feae12c9940>
```





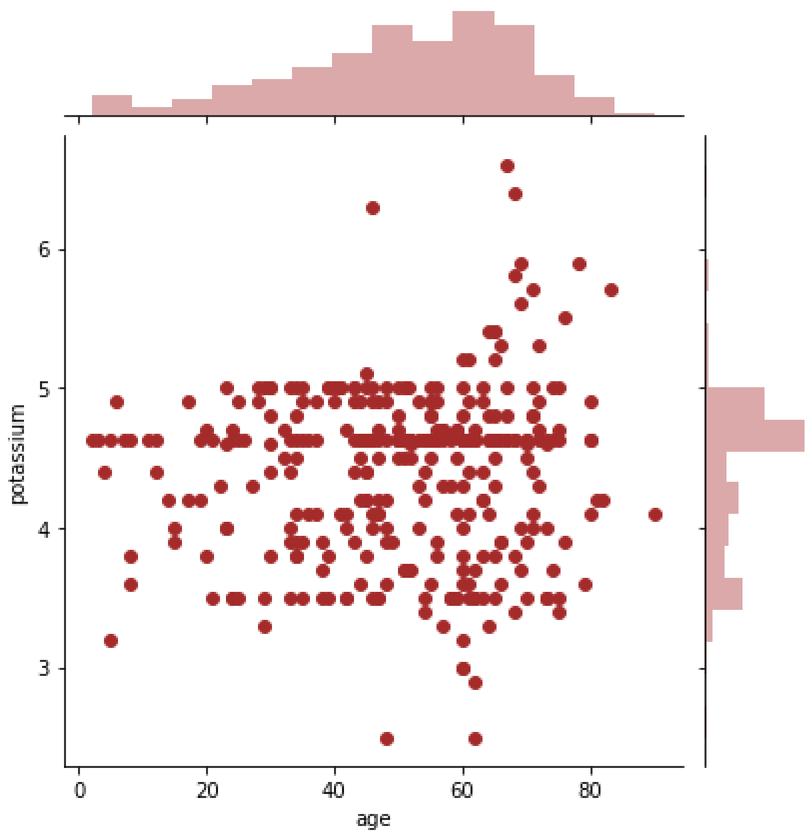
sodium
age

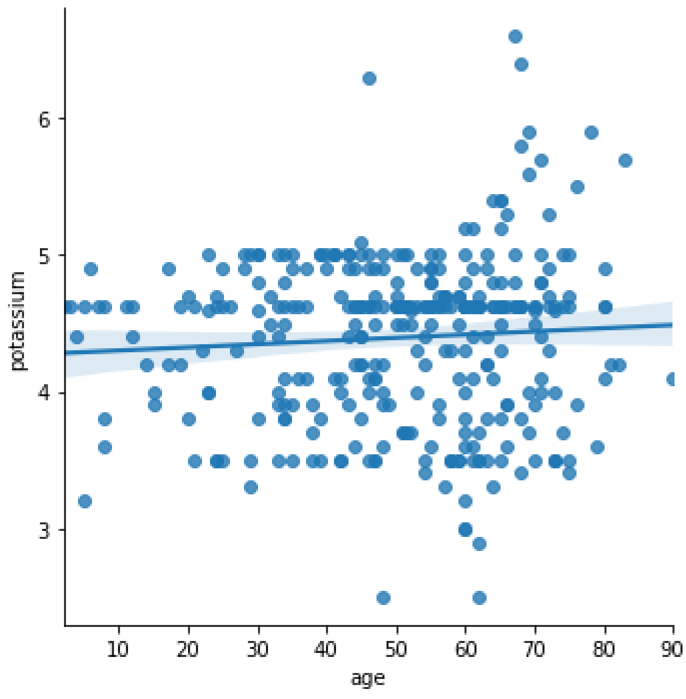
In [20]:

```
#Jointplot

sns.jointplot(x='age',y='potassium',data=df_imputed,color='brown')
sns.lmplot(x='age',y='potassium',data=df_imputed)

# blood_glucose_random blood_urea serum_creatinine sodium potassium
```

Out[20]:

```
<seaborn.axisgrid.FacetGrid at 0x7feae2ab96d0>
```



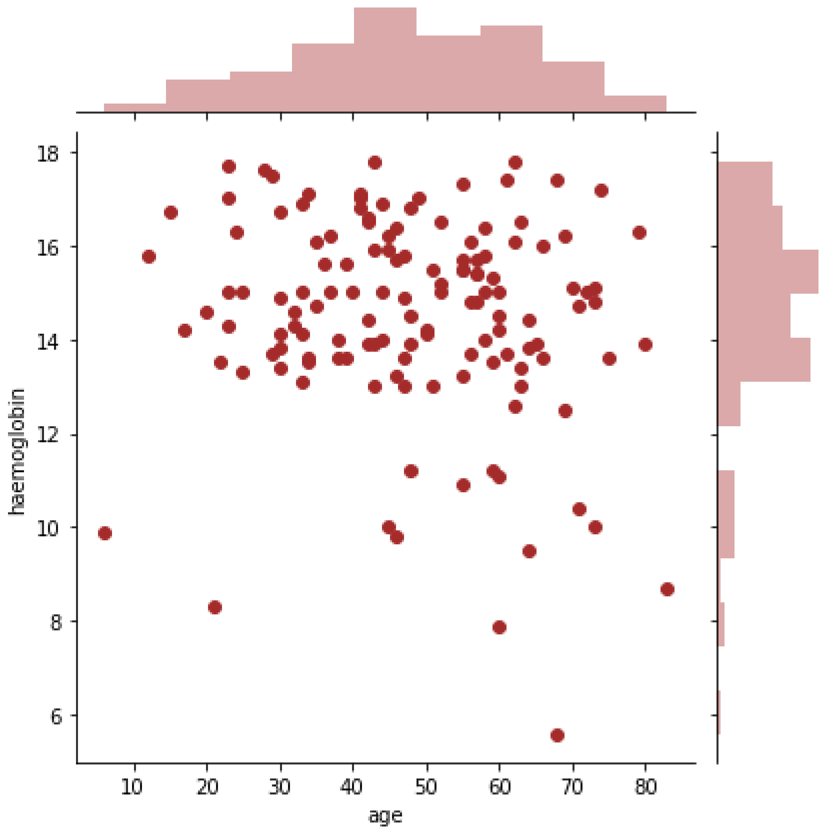


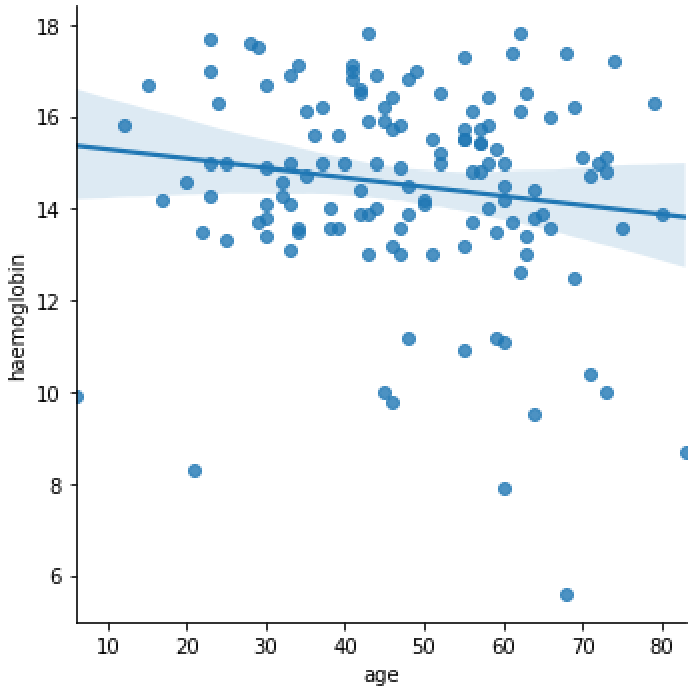
In [21]:

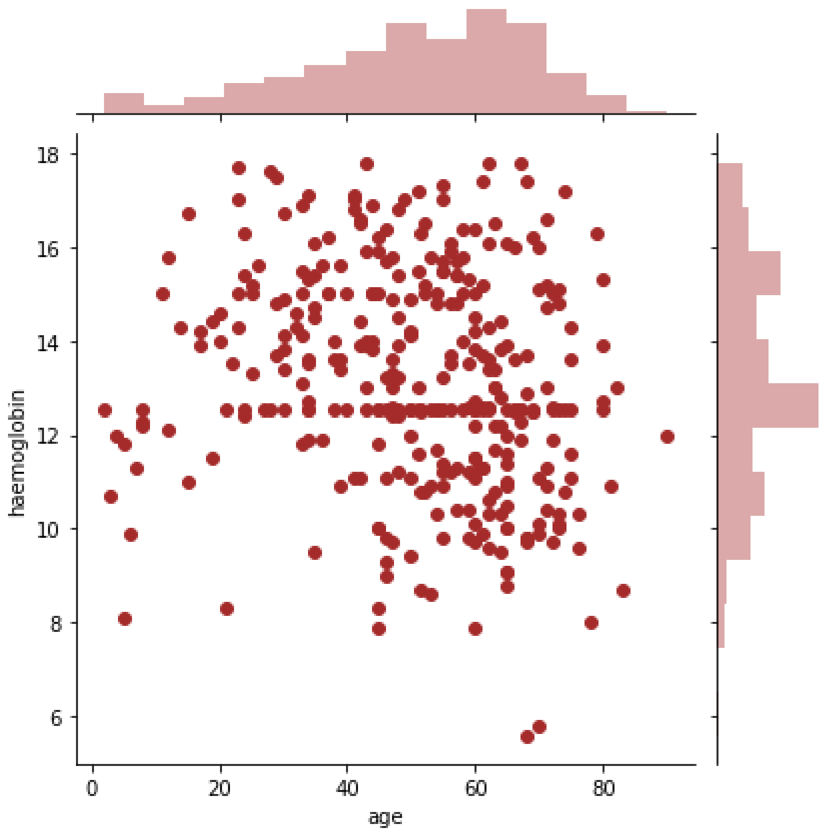
```
#Jointplot
sns.jointplot(x='age',y='haemoglobin',data=df_notnull,color='brown')
sns.lmplot(x='age',y='haemoglobin',data=df_notnull)

sns.jointplot(x='age',y='haemoglobin',data=df_imputed,color='brown')
sns.lmplot(x='age',y='haemoglobin',data=df_imputed)

# blood_glucose_random blood_urea serum_creatinine sodium potassium haemoglobin
```

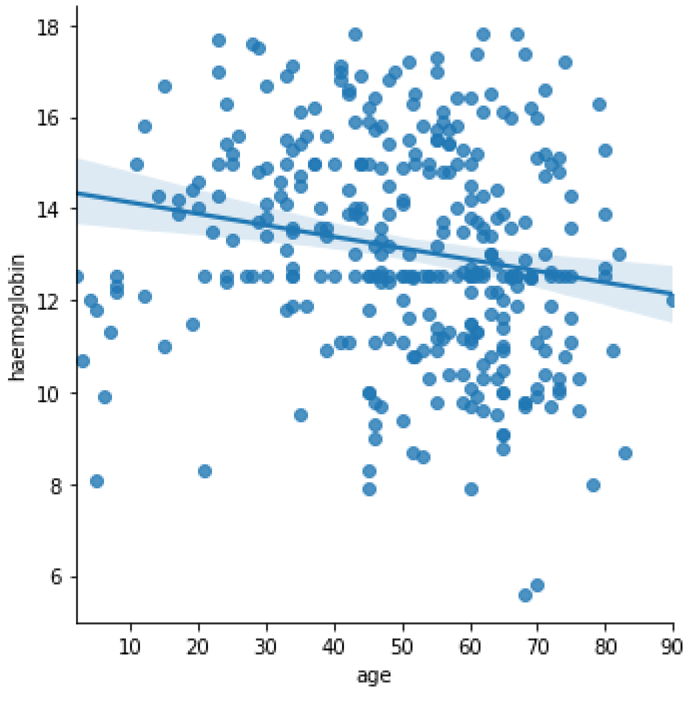
Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x7feae1542220>
```






haemoglobin
age
18
16
14
12
10
8
6
0
20
40
60
80

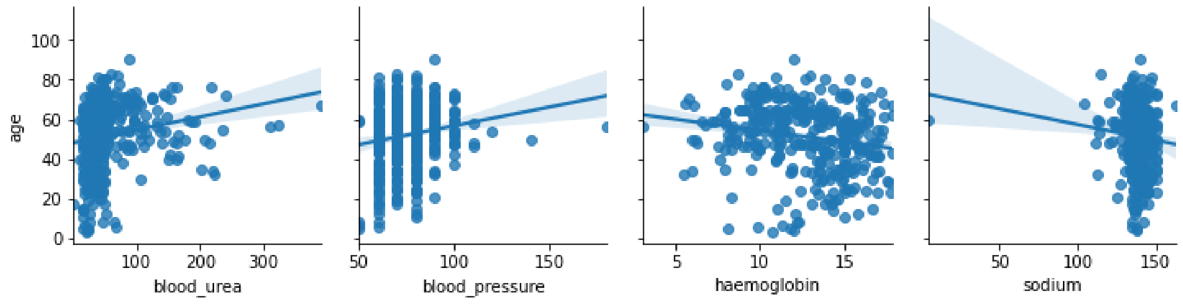

haemoglobin
age
18
16
14
12
10
8
6
10
20
30
40
50
60
70
80
90

In [22]:

```
sns.pairplot(df1,
             x_vars = ['blood_urea', 'blood_pressure',
                       'haemoglobin', 'sodium'],
             y_vars = 'age',
             kind   = "reg" )
```

Out[22]:

```
<seaborn.axisgrid.PairGrid at 0x7feae163d250>
```

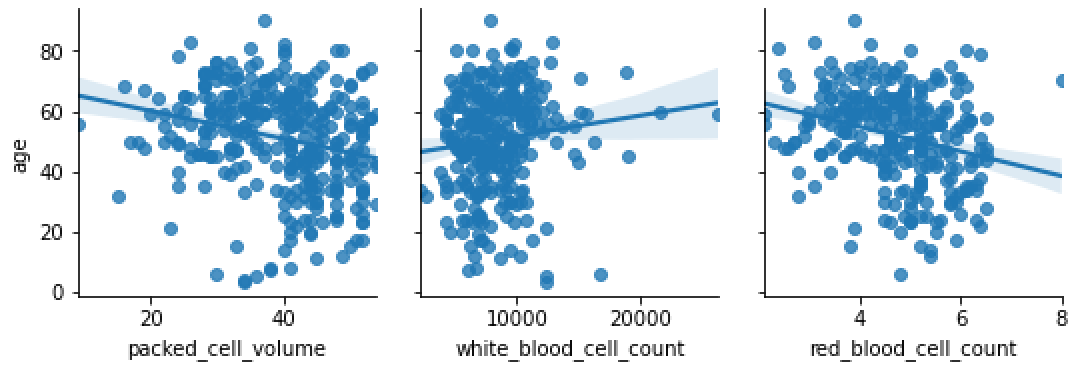


In [23]:

```
sns.pairplot(df1,
             x_vars = ['packed_cell_volume',
                        'white_blood_cell_count', 'red_blood_cell_count'],
             y_vars = 'age',
             kind   = "reg" )
```

Out[23]:

```
<seaborn.axisgrid.PairGrid at 0x7feae0e28700>
```



In [24]:

```
sns.pairplot(df_notnull)
```

In [25]:

```
sns.pairplot(df_imputed)
```

## Modelling

### Simple Linear Regression

In [94]:

```
# blood_glucose_random blood_urea serum_creatinine sodium potassium haemoglobin pack

X = df_notnull[["packed_cell_volume"]]
Y = df_notnull[['haemoglobin']]
```

In [95]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y) #, test_size=0.3, random_s

from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
```

Out[95]:

```
LinearRegression()
```

In [96]:

```
#Training Accuracy
linreg.score(X_train,y_train)

#Prediction
prediction=linreg.predict(X_test)

#Testing Accuracy
linreg.score(X_test,y_test)
```

Out[96]:

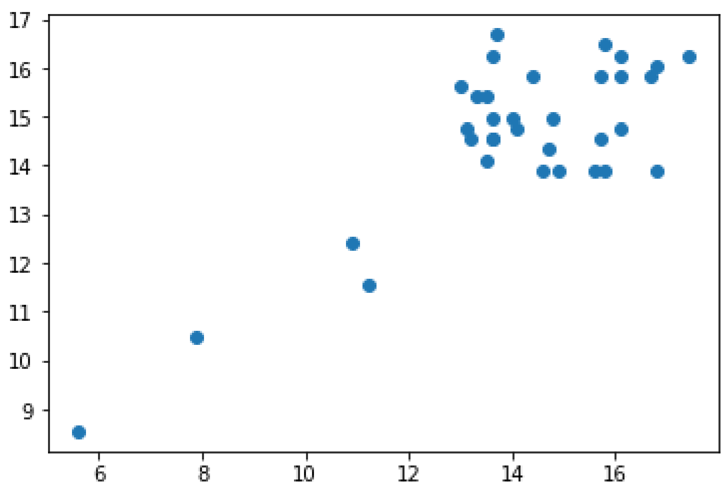
```
0.5701234680464349
```

In [97]:

```
plt.scatter(y_test,prediction)
```

Out[97]:

```
<matplotlib.collections.PathCollection at 0x7feae2aed760>
```



In [98]:

```
from sklearn import metrics
print("MAE: ",metrics.mean_absolute_error(y_test,prediction))
print("MSE: ",metrics.mean_squared_error(y_test,prediction))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE:  1.31760321006809
MSE:  2.454089260358441
RMSE:  1.5665533059422017
```

In [99]:

```
coef=pd.DataFrame()
coef['Features'] = X.columns.values
coef['Coefficients'] = linreg.coef_
coef
```

Out[99]:

| | Features | Coefficients |
|---|---|---|
| **0** | packed_cell_volume | 0.213976 |

In [100]:

```
#Testing Accuracy
linreg.score(X_test,y_test)
```

Out[100]:

```
0.5701234680464349
```

## Multiple Linear Regression

In [101]:

```
#Splitting into Training and Testing Data
X=df_notnull.drop(['age'],axis=1)
y=df_notnull[['red_blood_cell_count','potassium']]
X
```

Out[101]:

| | blood_pressure | blood_glucose_random | blood_urea | serum_creatinine | sodium | potassium |
|---|---|---|---|---|---|---|
| **3** | 70.0 | 117.0 | 56.0 | 3.8 | 111.0 | 2.5 |
| **14** | 80.0 | 157.0 | 90.0 | 4.1 | 130.0 | 6.4 |
| **27** | 70.0 | 264.0 | 87.0 | 2.7 | 130.0 | 4.0 |
| **48** | 70.0 | 70.0 | 32.0 | 0.9 | 125.0 | 4.0 |
| **71** | 60.0 | 163.0 | 92.0 | 3.3 | 141.0 | 4.0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **395** | 80.0 | 140.0 | 49.0 | 0.5 | 150.0 | 4.9 |
| **396** | 70.0 | 75.0 | 31.0 | 1.2 | 141.0 | 3.5 |
| **397** | 80.0 | 100.0 | 26.0 | 0.6 | 137.0 | 4.4 |
| **398** | 60.0 | 114.0 | 50.0 | 1.0 | 135.0 | 4.9 |
| **399** | 80.0 | 131.0 | 18.0 | 1.1 | 141.0 | 3.5 |

133 rows × 10 columns

In [102]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_stat
```

In [103]:

```
from sklearn.linear_model import LinearRegression

multi_lr=LinearRegression()
multi_lr.fit(X_train,y_train)

#Training Accuracy
multi_lr.score(X_train,y_train)
```

Out[103]:

```
1.0
```

In [104]:

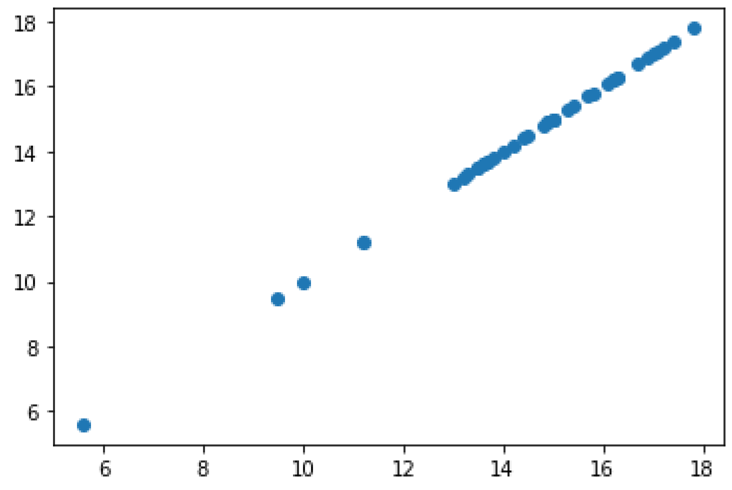
```
#Prediction
prediction=multi_lr.predict(X_test)
```

In [105]:

```
plt.scatter(y_test,prediction)
```

Out[105]:

```
<matplotlib.collections.PathCollection at 0x7feae160adf0>
```



In [106]:

```
from sklearn import metrics
print("MAE: ",metrics.mean_absolute_error(y_test,prediction))
print("MSE: ",metrics.mean_squared_error(y_test,prediction))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE:  2.7977620220553944e-15
MSE:  1.6487192919119146e-29
RMSE:  4.060442453615018e-15
```

In [107]:

```
#Splitting into Training and Testing Data
X=df_notnull.drop(['red_blood_cell_count','potassium','packed_cell_volume','blood_ur
y=df_notnull[['age']]
X
```

Out[107]:

| | age | blood_pressure | blood_glucose_random | serum_creatinine | sodium | haemoglobin | whit |
|---|---|---|---|---|---|---|---|
| **3** | 48.0 | 70.0 | 117.0 | 3.8 | 111.0 | 11.2 | |
| **14** | 68.0 | 80.0 | 157.0 | 4.1 | 130.0 | 5.6 | |
| **27** | 69.0 | 70.0 | 264.0 | 2.7 | 130.0 | 12.5 | |
| **48** | 73.0 | 70.0 | 70.0 | 0.9 | 125.0 | 10.0 | |
| **71** | 46.0 | 60.0 | 163.0 | 3.3 | 141.0 | 9.8 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **395** | 55.0 | 80.0 | 140.0 | 0.5 | 150.0 | 15.7 | |
| **396** | 42.0 | 70.0 | 75.0 | 1.2 | 141.0 | 16.5 | |
| **397** | 12.0 | 80.0 | 100.0 | 0.6 | 137.0 | 15.8 | |
| **398** | 17.0 | 60.0 | 114.0 | 1.0 | 135.0 | 14.2 | |
| **399** | 58.0 | 80.0 | 131.0 | 1.1 | 141.0 | 15.8 | |

133 rows × 7 columns

In [108]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_stat
```

In [109]:

```
from sklearn.linear_model import LinearRegression

multi_lr=LinearRegression()
multi_lr.fit(X_train,y_train)

#Training Accuracy
multi_lr.score(X_train,y_train)
```

Out[109]:

```
1.0
```

In [110]:

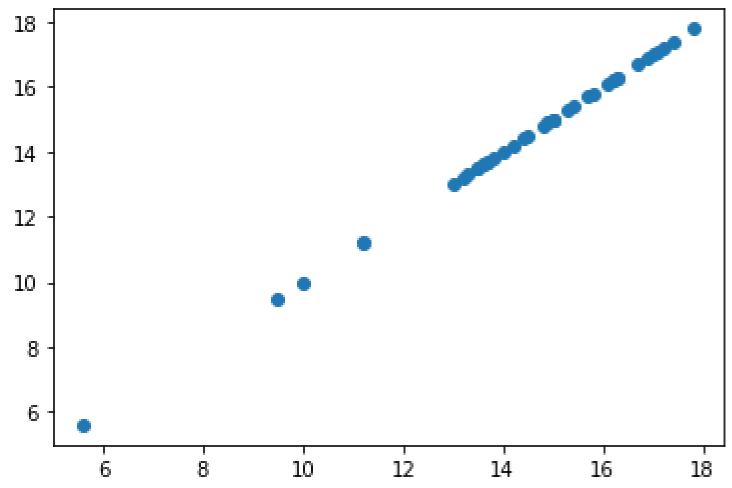
```
#Prediction
prediction=multi_lr.predict(X_test)
```

In [111]:

```
plt.scatter(y_test,prediction)
```

Out[111]:

```
<matplotlib.collections.PathCollection at 0x7feae2460f10>
```



In [112]:

```
from sklearn import metrics
print("MAE: ",metrics.mean_absolute_error(y_test,prediction))
print("MSE: ",metrics.mean_squared_error(y_test,prediction))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE:  2.930988785010413e-15
MSE:  2.6979042958558606e-29
RMSE:  5.194135438988726e-15
```

## Inference

Linear regression is one of the most common techniques of regression analysis. It is also called a simple linear regression. It establishes the relationship between two variables using a straight line. Linear regression attempts to draw a line that comes closest to the data by finding the slope and intercept that define the line and minimize regression errors.

Multiple regression is a broader class of regressions that encompasses linear and nonlinear regressions with multiple explanatory variables.

It is rare that a dependent variable is explained by only one variable. In this case, an analyst uses multiple regression, which attempts to explain dependent variable using more than one independent variable. Multiple regressions can be linear and nonlinear.

In [ ]:

```

```