Open in Colab

Group307_kyphosis

1. Name - Arindam Dey
2. Group - 307 Group Members - ARINDAM DEY - 2020fc04251@wilp.bits-pilani.ac.in KAUSHIK DUBEY - 2020fc04245@wilp.bits-pilani.ac.in MOHAMMAD ATTAULLAH - 2020FC04274@wilp.bits-pilani.ac.in

3. Dataset Name - kyphosis.csv https://github.com/arindamdeyofficial/vertebraeSurgeryRiskFactor/blob/main/kyphosis.csv

RAW- https://raw.githubusercontent.com/arindamdeyofficial/vertebraeSurgeryRiskFactor/7e5ba322ac93a5636528e286894521d35f7d4995/kyphosis.c

1. Import Libraries/Dataset

b. Import the required libraries

In [126…

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

In [127…

```
df = pd.read_csv('https://raw.githubusercontent.com/arindamdeyofficial/vertebraeSurgeryRiskFactor/7e5ba322ac93a
```

1. Data Visualization and Exploration

a. Print at least 5 rows for sanity check to identify all the features present in the dataset and if the target matches with them.

In [30]:

```
df.head()
```

Out[30]:

	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Kyphosis  81 non-null     object
 1   Age       81 non-null     int64 
 2   Number    81 non-null     int64 
 3   Start     81 non-null     int64 
dtypes: int64(3), object(1)
memory usage: 2.7+ KB
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['Kyphosis', 'Age', 'Number', 'Start'], dtype='object')
```

1. Data Visualization and Exploration

b. Print the description and shape of the dataset.

In [6]:

```
df.describe()
```

Out[6]:

	Age	Number	Start
count	81.000000	81.000000	81.000000
mean	83.654321	4.049383	11.493827
std	58.104251	1.619423	4.883962
min	1.000000	2.000000	1.000000
25%	26.000000	3.000000	9.000000
50%	87.000000	4.000000	13.000000
75%	130.000000	5.000000	16.000000
max	206.000000	10.000000	18.000000

In [7]:

```
df.shape
```

Out[7]:

```
(81, 4)
```

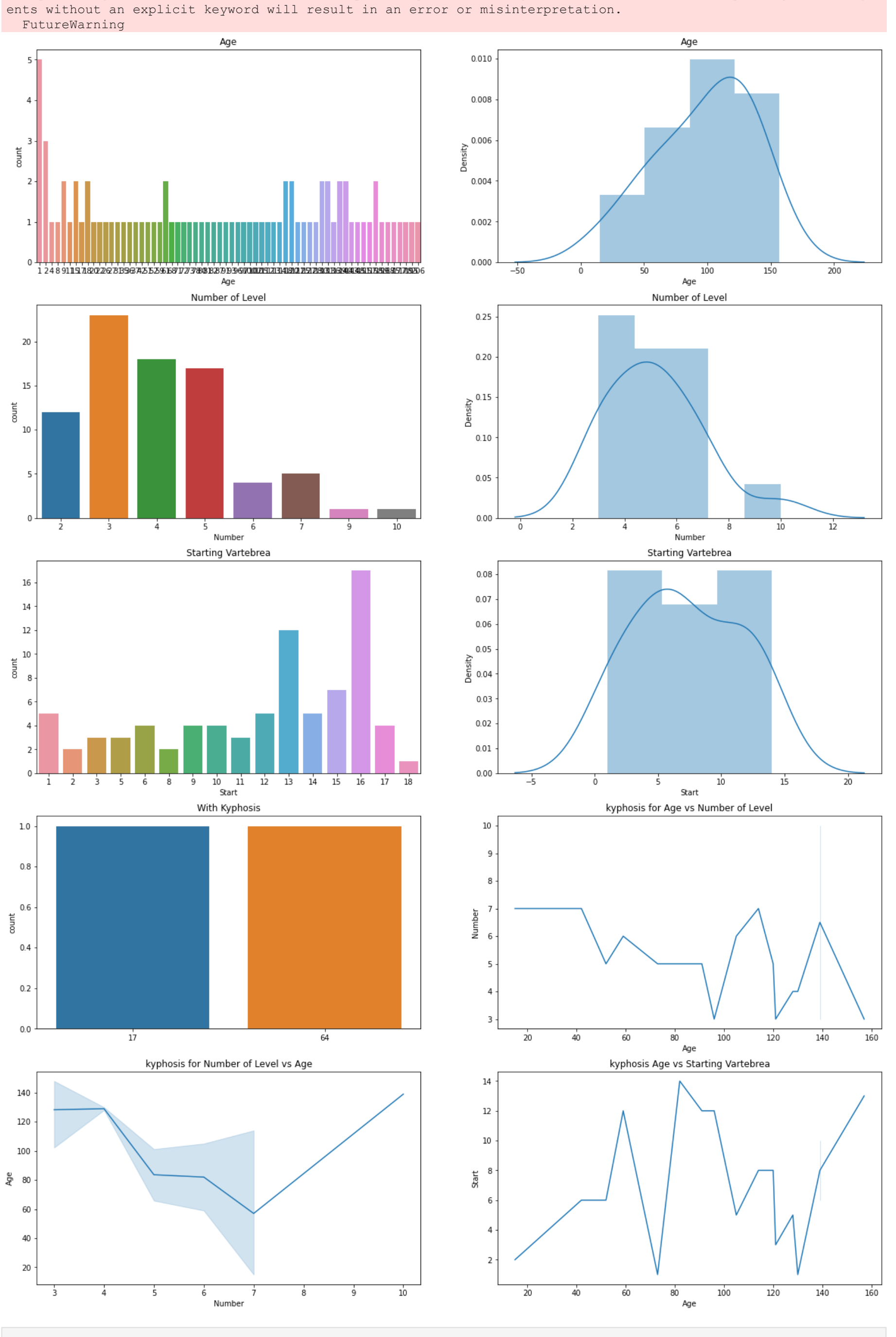
1. Data Visualization and Exploration

c. Provide appropriate visualization to get an insight about the dataset.

In [8]:

```
#Number === number of levels involved
#Start === starting vertebrae level of the surgery

n_rows=5
n_cols=2
width=20
height=30

fig,ax=plt.subplots(n_rows,n_cols,sharex=False,sharey=False,figsize=(width,height))
#Data normalisation view
wk = df[df["Kyphosis"]=="present"]

#Age
g = sns.countplot(df["Age"], ax=ax[0,0])
ax[0,0].set_title("Age")

g = sns.distplot(wk["Age"], ax=ax[0,1])
ax[0,1].set_title("Age")

#Number
g = sns.countplot(df["Number"], ax=ax[1,0])
ax[1,0].set_title("Number of Level")

g = sns.distplot(wk["Number"], ax=ax[1,1])
ax[1,1].set_title("Number of Level")

#Start
g = sns.countplot(df["Start"], ax=ax[2,0])
ax[2,0].set_title("Starting Vartebrea")

g = sns.distplot(wk["Start"], ax=ax[2,1])
ax[2,1].set_title("Starting Vartebrea")

#count kyphosis
l = len(df[df["Kyphosis"]=="present"])
g = sns.countplot([l,len(df)-l], ax=ax[3,0])
ax[3,0].set_title("With Kyphosis")

#Count kyphosis for Age vs Number of Level
#Having kyphosis Age vs Number Lineplot Age in X axis
wk = df[df["Kyphosis"]=="present"]
g = sns.lineplot(data=wk, x="Age", y="Number", ax=ax[3,1])
ax[3,1].set_title("kyphosis for Age vs Number of Level")

#Having kyphosis Number vs Age Countplot Number in X axis
g = sns.lineplot(data=wk, x="Number", y="Age", ax=ax[4,0])
ax[4,0].set_title("kyphosis for Number of Level vs Age")

#Having kyphosis Age vs Starting Vartebrea
g = sns.lineplot(data=wk, x="Age", y="Start",  ax=ax[4,1])
ax[4,1].set_title("kyphosis Age vs Starting Vartebrea")

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as
a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as
a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as
a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as
a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

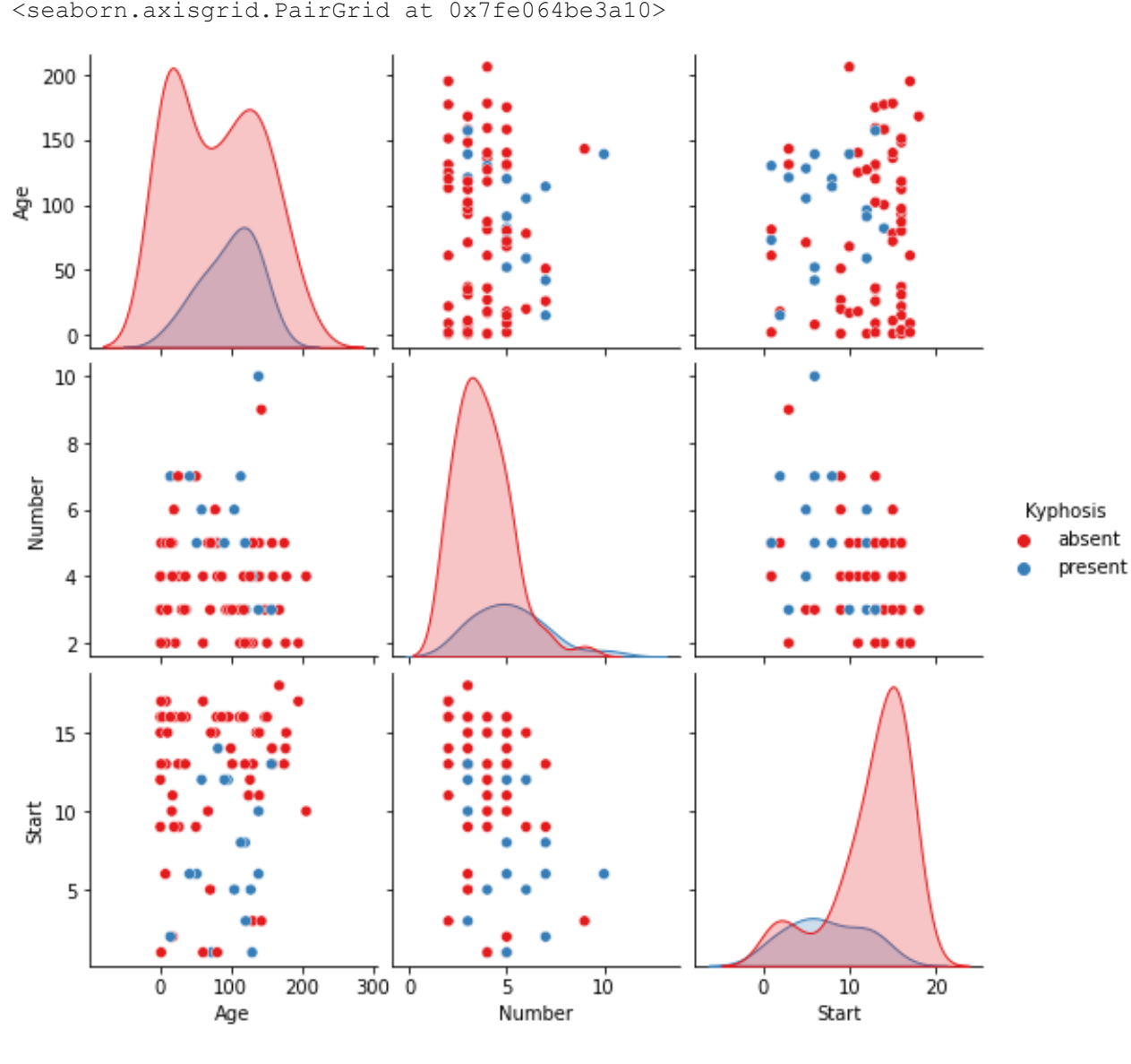


In [9]:

```
sns.pairplot(df,hue='Kyphosis',palette='Set1')
```

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x7fe064be3a10>
```



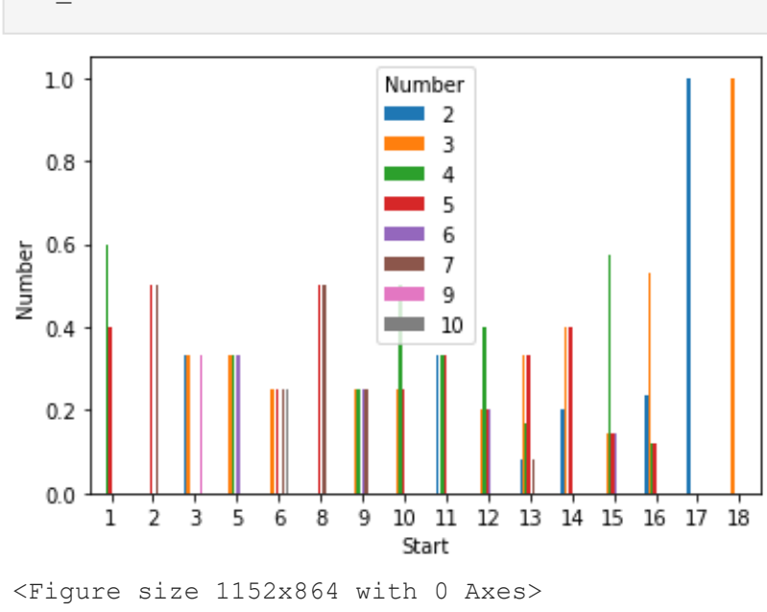
BiVariate analysis

In [10]:

```
def bi_variate(df, col1, col2):
    cross_name=pd.crosstab(df[col1], df[col2], normalize='index')
    cross_name.plot.bar()
    plt.xlabel('{}'.format(col1))
    plt.ylabel('{}'.format(col2))
    plt.xticks(rotation=0)
    plt.figure(figsize=(16,12))
    plt.show()
    return cross_name*100
```

In [ ]:

```
bi_variate(df, 'Start', 'Number')
```



```
<Figure size 1152x864 with 0 Axes>
```

Out[ ]:

Number	2	3	4	5	6	7	9	10
Start								
1	0.000000	0.000000	60.000000	40.000000	0.000000	0.000000	0.000000	0.0
2	0.000000	0.000000	0.000000	50.000000	0.000000	50.000000	0.000000	0.0
3	33.333333	33.333333	0.000000	0.000000	0.000000	0.000000	33.333333	0.0
5	0.000000	33.333333	33.333333	0.000000	33.333333	0.000000	0.000000	0.0
6	0.000000	25.000000	0.000000	25.000000	0.000000	25.000000	0.000000	25.0
8	0.000000	0.000000	0.000000	50.000000	0.000000	50.000000	0.000000	0.0
9	0.000000	25.000000	25.000000	0.000000	25.000000	25.000000	0.000000	0.0
10	0.000000	25.000000	50.000000	25.000000	0.000000	0.000000	0.000000	0.0
11	33.333333	0.000000	33.333333	33.333333	0.000000	0.000000	0.000000	0.0
12	0.000000	20.000000	40.000000	20.000000	20.000000	0.000000	0.000000	0.0
13	8.333333	33.333333	16.666667	33.333333	0.000000	8.333333	0.000000	0.0
14	20.000000	40.000000	0.000000	40.000000	0.000000	0.000000	0.000000	0.0
15	0.000000	14.285714	57.142857	14.285714	14.285714	0.000000	0.000000	0.0
16	23.529412	52.941176	11.764706	11.764706	0.000000	0.000000	0.000000	0.0
17	100.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
18	0.000000	100.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0

d. Try exploring the data and see what insights can be drawn from the dataset.

Exploratory data analysis

1. Data Pre-processing and cleaning

a. Do the appropriate preprocessing of the data like identifying NULL or Missing Values if any, handling of outliers if present in the dataset, skewed data etc. Apply appropriate feature engineering techniques for them.

Now First check should be any Null is present or not

In [11]:

```
df.isna().any()
```

Out[11]:

```
Kyphosis    False
Age         False
Number      False
Start       False
dtype: bool
```

In [12]:

```
df.isna().sum()
```

Out[12]:

```
Kyphosis    0
Age         0
Number      0
Start       0
dtype: int64
```

Now there are no Null values present

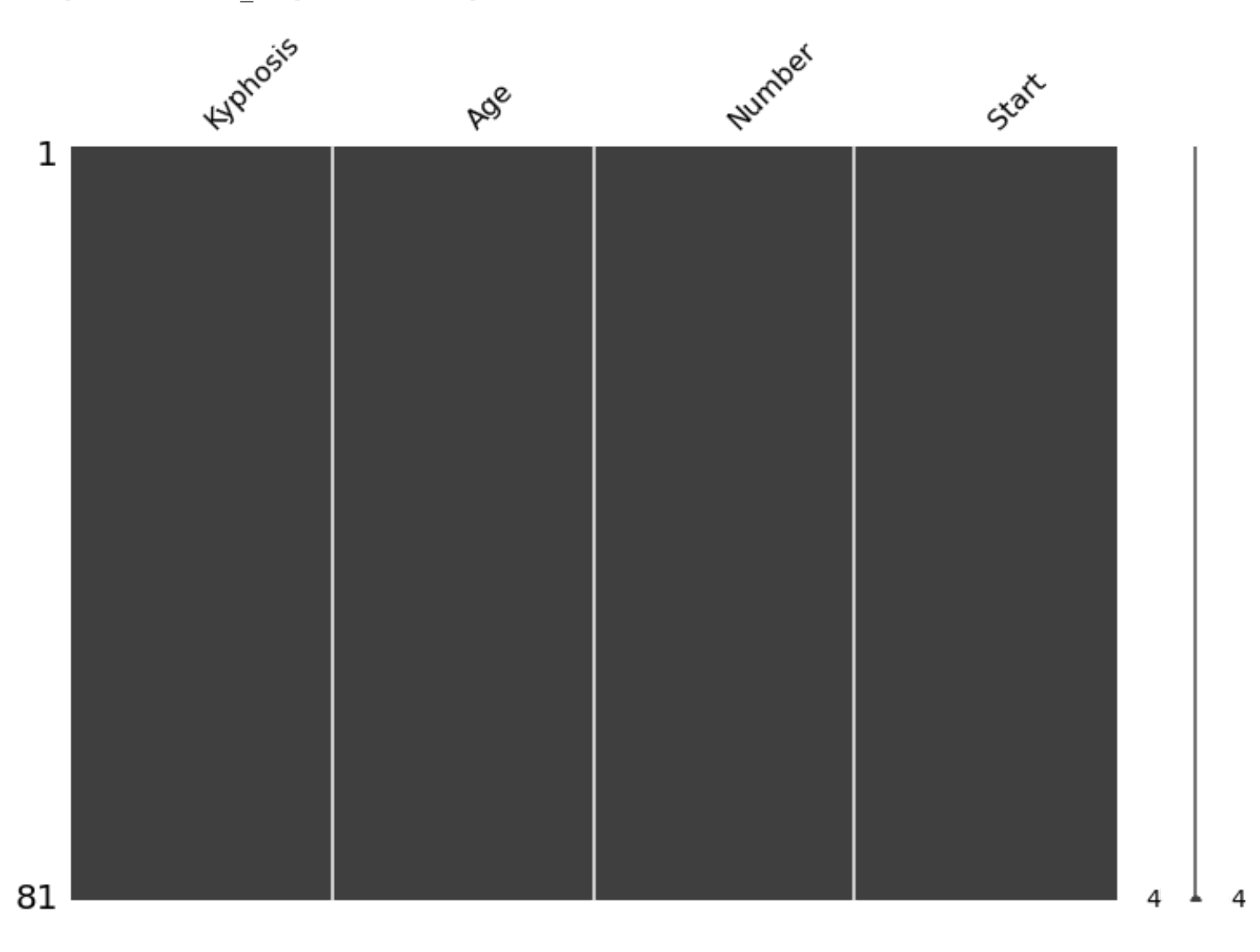
In [13]:

```
!pip install missingno
import missingno
missingno.matrix(df,figsize=(12,8))
```

```
Requirement already satisfied: missingno in /usr/local/lib/python3.7/dist-packages (0.5.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from missingno) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from missingno) (1.19.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from missingno) (3.2.2)
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (from missingno) (0.11.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib-
>missingno) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packag
es (from matplotlib->missingno) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->missing
no) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mi
ssingno) (1.3.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib->mi
ssingno) (1.15.0)
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.7/dist-packages (from seaborn->missingno)
(1.1.5)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.23->seabo
rn->missingno) (2018.9)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe061dfcad0>
```



Outlier Identificayion and removal

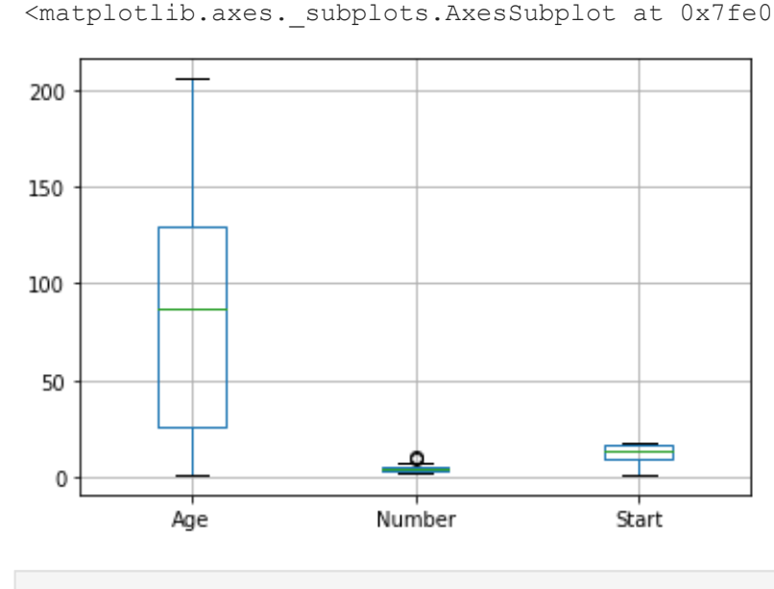
Box plot use the Inter Quantile Range(IQR) method. This is easier so will be using

In [14]:

```
df.boxplot()
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe057ae50d0>
```

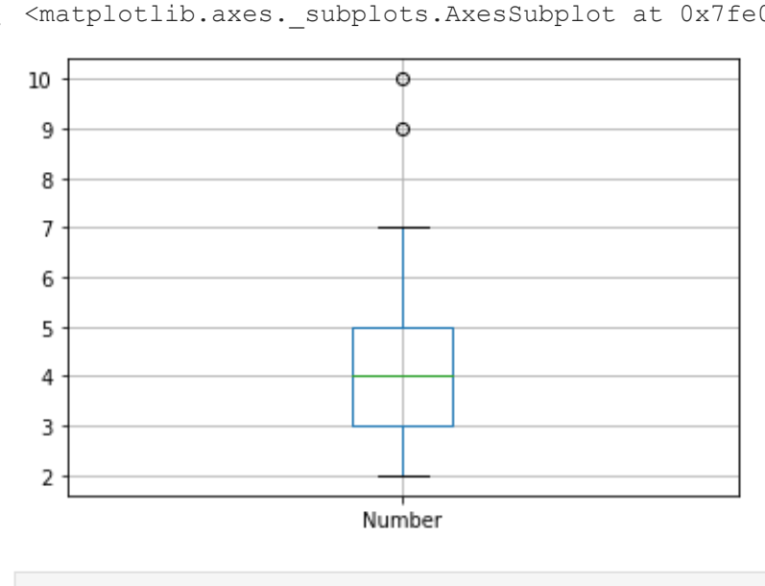


In [128…

```
df.boxplot(column=['Number'])
```

Out[128…

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0529716d0>
```

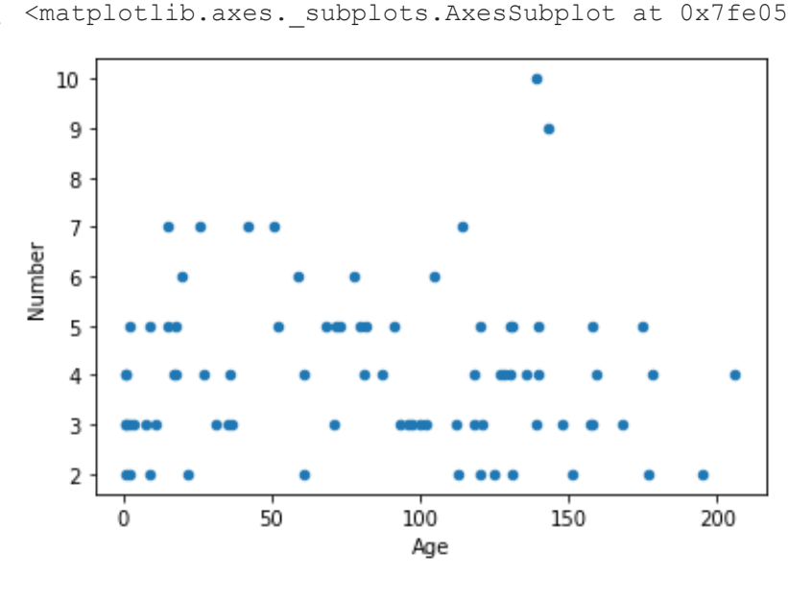


In [129…

```
df.plot.scatter('Age', 'Number')
```

Out[129…

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe05295ad50>
```



Z-score is the signed number of standard deviations by which the value of an observation or data point is above the mean value of what is being observed or measured Through thi method also we can identify Outlier

In [130…

```
mean = np.mean(df["Number"])
mean
```

Out[130…

```
4.049382716049383
```

In [131…

```
std = np.std(df["Number"])
std
```

Out[131…

```
1.6093955200472556
```

In [118…

```
threshold = 2
outlierCombo = []
outlier = []
zScore = []
for i in df["Number"]:
    z = (i-mean)/std
    if z > threshold:
      outlierCombo.append([i,z])
      outlier.append(i)
      zScore.append(z)
outlierCombo
```

Out[118…

```
[[9, 3.076072489505411], [10, 3.697423790378075]]
```

In [132…

```
df[df["Number"]>=9]
```

Out[132…

	Kyphosis	Age	Number	Start
42	absent	143	9	3
52	present	139	10	6

This 2 are really unrealistic Age people live according to age also and as per scatter plot and ZScore. But data is skewed toward 100++ years of age and data quantity also less so prefferable not drop but replace in 3 sigma method the 2 outliers.

In [133…

```
leftVal = mean - 3*std
rightVal = mean + 3*std
print(mean, std, leftVal,rightVal)
outlL = df[df["Number"]<leftVal]
outlR = df[df["Number"]> rightVal]
print(outlL["Number"].count(), outlR["Number"].count())
print("threshold value count: ", df[df["Number"] == rightVal]["Number"].count())
print(outlR)
```

```
4.049382716049383 1.6093955200472556 -0.7788038440923835 8.87756927619115
0 2
threshold value count:  0
   Kyphosis  Age  Number  Start
42   absent  143       9      3
52  present  139      10      6
```

In [134…

```
df.loc[df["Number"] > rightVal, "Number"] = rightVal
```

In [135…

```
df[df["Number"] > rightVal]
```

Out[135…

Kyphosis	Age	Number	Start

In [136…

```
print("threshold value count: ", df[df["Number"] == rightVal]["Number"].count())
```

```
threshold value count:  2
```

In [137…

```
index_names = df.loc[df["Number"]>=9].index
index_names
```

Out[137…

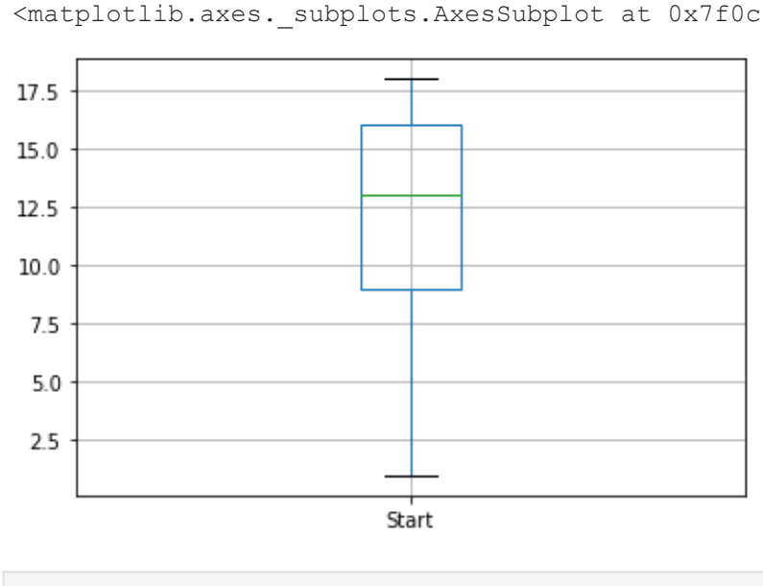
```
Int64Index([], dtype='int64')
```

In [ ]:

```
df.boxplot(column=['Start'])
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0c9de6c4d0>
```



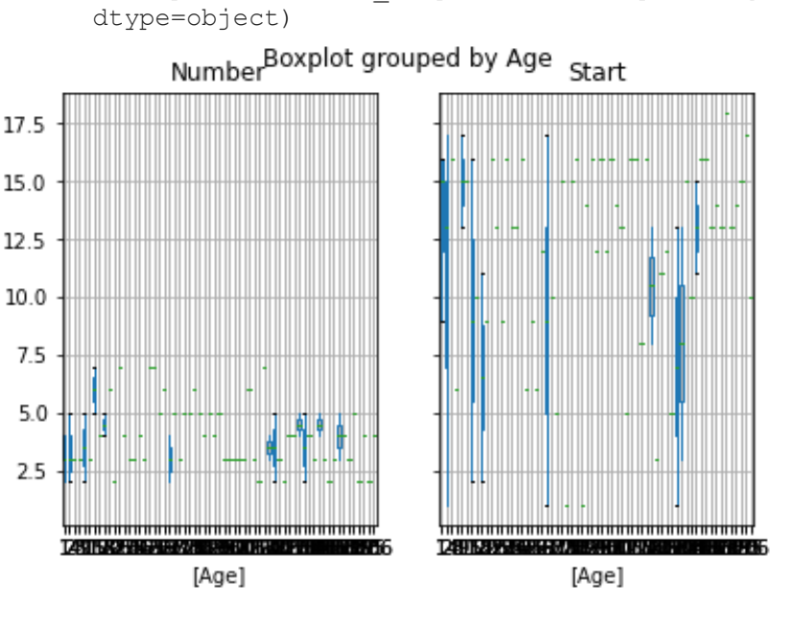
In [ ]:

```
df.boxplot(by="Age")
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarra
y from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarra
y from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
```

Out[ ]:

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f0c86e6a6d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f0c86e21550>],
      dtype=object)
```



b. Apply the feature transformation techniques like Standardization, Normalization, etc. You are free to apply the appropriate transformations depending upon the structure and the complexity of your dataset.

Replace categorical Column with Numeric value

Column segregartion

In [138…

```
df.head(10)
#Encoding (categorical to Numerical)
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

#target column
target_col=['Kyphosis']

#categorical cols
categorical_cols=df.nunique()[df.nunique() < 5].keys().tolist()
categorical_cols=[x for x in categorical_cols if x not in target_col]
print("categorical columns: {}".format(categorical_cols))

#numerical cols
cont_cols=[x for x in df.columns if x not in categorical_cols+target_col]
print("Numerical columns: {}".format(cont_cols))

#binary cols
bin_cols=df.nunique()[df.nunique() == 2].keys().tolist()
print("Binary columns: {}".format(bin_cols))

#multi cols(more than 2 values)
mul_cols=[x for x in categorical_cols if x not in bin_cols]
print("Multi valued columns: {}".format(mul_cols))
```

```
categorical columns: []
Numerical columns: ['Age', 'Number', 'Start']
Binary columns: ['Kyphosis']
Multi valued columns: []
```

In [139…

```
#Label Encoding for Binary cols
le=LabelEncoder()
for i in bin_cols:
    df[i]=le.fit_transform(df[i])

#Duplicating cols for multi-value cols
#df=pd.get_dummies(data=df,columns=mul_cols)
df.head(10)
```

Out[139…

	Kyphosis	Age	Number	Start
0	0	71	3.0	5
1	0	158	3.0	14
2	1	128	4.0	5
3	0	2	5.0	1
4	0	1	4.0	15
5	0	1	2.0	16
6	0	61	2.0	17
7	0	37	3.0	16
8	0	113	2.0	16
9	1	59	6.0	12

1. Data Pre-processing and cleaning

c. Do the correlational analysis on the dataset. Provide a visualization for the same.

Feature Selection

We can do by corelation. mostly continuous data as input and categorical data as output. SO we can use below methods.

1. ANOVA correlation coefficient (linear)
2. Kendall's rank coefficient (nonlinear)

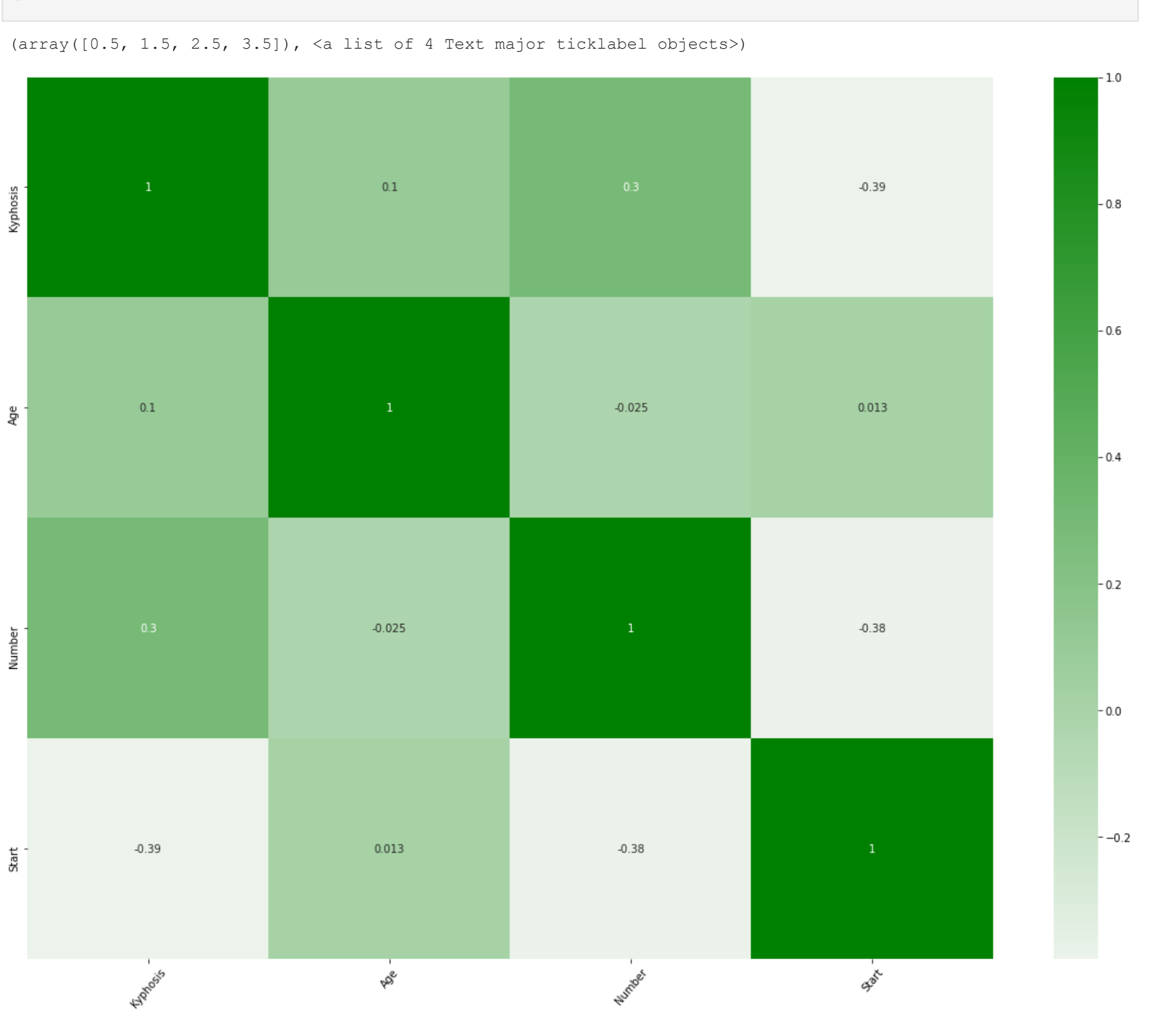
kendall

In [140…

```
#Co-relation
#correlation analysis shows us how to determine both the nature and strength of relationship between two variab
#correaltion lies between -1 to 1 (0: No corelation; -1: perfect negative correlation; +1: positive core)
correlation=df.corr(method='kendall')
correlation

#plotting correlation Matrix
plt.figure(figsize=(20,15))
cm = sns.light_palette("green", as_cmap=True)
sns.heatmap(correlation, xticklabels=correlation.columns.values, yticklabels=correlation.columns.values, cmap=c
plt.xticks(rotation=50)
```

Out[140…

```
(array([0.5, 1.5, 2.5, 3.5]), <a list of 4 Text major ticklabel objects>)
```



Corelation between coloumns are not between -1 to + 1 i.e. the best but not greater than 0.5 also and between kyphosis as target and other column is generally greater. so we are good.

ANOVA F measure

In [142…

```
# ANOVA feature selection for numeric input and categorical output
from sklearn.datasets import make_classification
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
# generate dataset
X, y = make_classification(n_samples=100, n_features=20, n_informative=2)
# define feature selection
fs = SelectKBest(score_func=f_classif, k=2)
# apply feature selection
X_selected = fs.fit_transform(X, y)
print(X_selected.shape)
```

```
(100, 2)
```

1. Data Preparation

a. Do the final feature selection and extract them into Column X and the class label into Column into Y.

In [ ]:

```
x=df.drop('Kyphosis',axis=1)
y=df['Kyphosis']
```

1. Data Preparation

b. Split the dataset into training and test sets.

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=101)
```

Part B

1. Model Building

a. Perform Model Development using at least three models, separately. You are free to apply any Machine Learning Models on the dataset. Deep Learning Models are strictly not allowed.

We will try Descision Tree as data is too small and number of column is also less preferrebly. And also try Random forest and Logistics regression as all of them are Classification models.

We have general common classification techniques

1. Logistic Regression
2. Naïve Bayes
3. Stochastic Gradient Descent
4. K-Nearest Neighbours
5. Decision Tree
6. Random Forest
7. Support Vector Machine

1. Model Building

DecisionTree

In [ ]:

```
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier()
```

b. Train the model and print the training accuracy and loss values

In [ ]:

```
dtree.fit(x_train,y_train)
```

Out[ ]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, 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='deprecated',
                       random_state=None, splitter='best')
```

1. Performance Evaluation

b. Do the prediction for the test data and display the results for the inference

In [ ]:

```
prediction=dtree.predict(x_test)
```

c. Print the confusion matrix. Provide appropriate analysis for the same.

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(y_test,y_pred))
```

```
print(classification_report(y_test,y_pred2))
```

```
              precision    recall  f1-score   support

           0       0.78      0.97      0.86       124
           1       0.94      0.67      0.78        76

    accuracy                           0.83       200
   macro avg       0.86      0.82      0.82       200
weighted avg       0.85      0.83      0.82       200
```

Hyperplane

```
from sklearn.svm import SVC,LinearSVC,NuSVC
```

```
clf=SVC(kernel='linear',random_state=0)
```

```
clf.fit(X_train,y_train)
```

```
SVC(kernel='linear', random_state=0)
```

```
y2_pred=clf.predict(X_test)
```

```
print(classification_report(y_test,y2_pred))
```

```
print(classification_report(y_test,y2_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.97      0.86       124
           1       0.94      0.67      0.78        76

    accuracy                           0.83       200
   macro avg       0.86      0.82      0.82       200
weighted avg       0.85      0.83      0.82       200
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

```
lr=LogisticRegression(random_state=0,solver='liblinear')
```

```
lr.fit(X_train,y_train)
```

```
LogisticRegression(random_state=0, solver='liblinear')
```

```
y3_pred=lr.predict(X_test)
```

```
print(classification_report(y_test,y3_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.97      0.86       124
           1       0.94      0.67      0.78        76

    accuracy                           0.83       200
   macro avg       0.86      0.82      0.82       200
weighted avg       0.85      0.83      0.82       200
```

```
from sklearn.model_selection import cross_val_score,GridSearchCV,RandomizedSearchCV
from sklearn.svm import SVC,LinearSVC,NuSVC
from sklearn.tree import DecisionTreeClassifier,ExtraTreeClassifier
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier,AdaBoostClassifier,GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB,BernoulliNB
from sklearn.linear_model import LogisticRegression,SGDClassifier,PassiveAggressiveClassifier,RidgeClassifier,RidgeClassifierCV,Perceptron
```

```
cv=GridSearchCV(estimator=lr, n_jobs=-1,verbose=1,cv=5,
                param_grid={'penalty': ['l1', 'l2', 'elasticnet', 'none'],
                            'dual': [True, False],
                            'tol': [0.0001, 0.0002, 0.0003, 0.0004, 0.0005],
                            'C': [1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9],
                            'fit_intercept': [True, False],
                            'intercept_scaling': [1, 2, 3, 4, 5, 6, 7, 8, 9],
                            'class_weight': ['balanced', None],
                            'random_state': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
                            'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
                            'max_iter': [100, 200, 300, 400, 500],
                            'multi_class': ['auto', 'ovr', 'multinomial'],
                            'warm_start': [True, False],
                            'l1_ratio': [None, 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]},
                scoring='accuracy')
```

```
cv.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 1440 candidates, totalling 7200 fits
```

```
cv.best_params_,cv.best_score_
```

```
({'C': 1.0,
  'class_weight': None,
  'dual': False,
  'fit_intercept': True,
  'intercept_scaling': 1,
  'l1_ratio': None,
  'multi_class': 'auto',
  'penalty': 'l2',
  'random_state': 0,
  'solver': 'newton-cg',
  'tol': 0.0001,
  'warm_start': False},
 0.8125)
```

```
y_pred=cv.predict(X_test)
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.97      0.86       124
           1       0.94      0.67      0.78        76

    accuracy                           0.83       200
   macro avg       0.86      0.82      0.82       200
weighted avg       0.85      0.83      0.82       200
```

```
from sklearn.metrics import cohen_kappa_score,matthews_corrcoef,confusion_matrix
```

```
print(cohen_kappa_score(y_test,y_pred))
```

```
0.6511111111111111
```

```
print(matthews_corrcoef(y_test,y_pred))
```

```
0.6711111111111111
```

```
print(confusion_matrix(y_test,y_pred))
```

```
[[120   4]
 [ 25  51]]
```