Problem statement: Performing exploratory data analysis using iris dataset

```
#import the useful libraries.
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Read the data set of "Iris" in data.
df= pd.read_csv("Iris dataset.csv")

# Printing the data
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
#print the head of the data frame.
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
df.shape
```

```
(150, 6)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
df.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000

```
df.isnull().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
data = df.drop_duplicates(subset ="Species",)
data
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
50	51	7.0	3.2	4.7	1.4	Iris-versicolor
100	101	6.3	3.3	6.0	2.5	Iris-virginica

```
df.value_counts("Species")
```

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
```

```
sns.countplot(x='Species', data=df, )
plt.show()
```

```
sns.scatterplot(x='SepalLengthCm', y='SepalWidthCm',
                hue='Species', data=df, )

# Placing Legend outside the Figure
```

```
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()


# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.scatterplot(x='PetalLengthCm', y='PetalWidthCm',
                hue='Species', data=df, )

# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()




# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.pairplot(df.drop(['Id'], axis = 1),
             hue='Species', height=2)




# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


fig, axes = plt.subplots(2, 2, figsize=(10,10))

axes[0,0].set_title("Sepal Length")
axes[0,0].hist(df['SepalLengthCm'], bins=7)

axes[0,1].set_title("Sepal Width")
axes[0,1].hist(df['SepalWidthCm'], bins=5);

axes[1,0].set_title("Petal Length")
axes[1,0].hist(df['PetalLengthCm'], bins=6);

axes[1,1].set_title("Petal Width")
axes[1,1].hist(df['PetalWidthCm'], bins=6);




# importing packages
import seaborn as sns
import matplotlib.pyplot as plt

plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "SepalLengthCm").add_legend()
```

```
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "SepalWidthCm").add_legend()

plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "PetalLengthCm").add_legend()

plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "PetalWidthCm").add_legend()

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
```

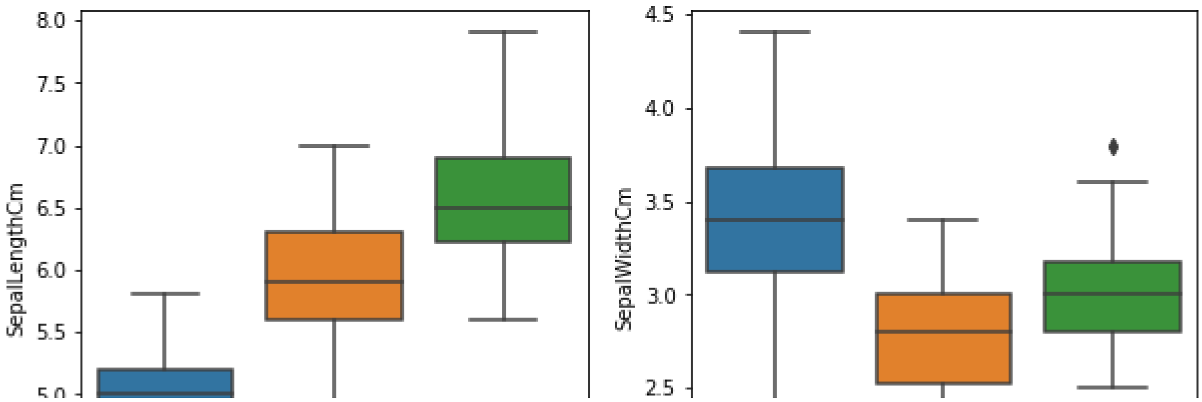
```
data.corr(method='pearson')
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.624413	-0.654654	0.969909	0.999685
SepalLengthCm	0.624413	1.000000	-0.999226	0.795795	0.643817
SepalWidthCm	-0.654654	-0.999226	1.000000	-0.818999	-0.673417
PetalLengthCm	0.969909	0.795795	-0.818999	1.000000	0.975713
PetalWidthCm	0.999685	0.643817	-0.673417	0.975713	1.000000

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.heatmap(df.corr(method='pearson').drop(
  ['Id'], axis=1).drop(['Id'], axis=0),
            annot = True);

plt.show()
```

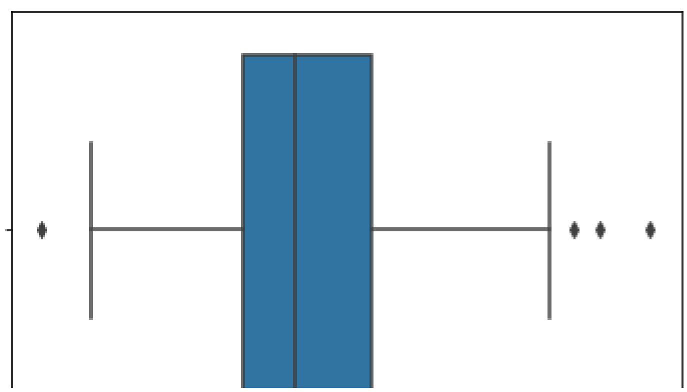
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt

def graph(y):
    sns.boxplot(x="Species", y=y, data=df)

plt.figure(figsize=(10,10))

# Adding the subplot at the specified
# grid position
plt.subplot(221)
graph('SepalLengthCm')

plt.subplot(222)
```

```
graph('SepalWidthCm')

plt.subplot(223)
graph('PetalLengthCm')

plt.subplot(224)
graph('PetalWidthCm')

plt.show()
```




```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('Iris dataset.csv')

sns.boxplot(x='SepalWidthCm', data=df)
```

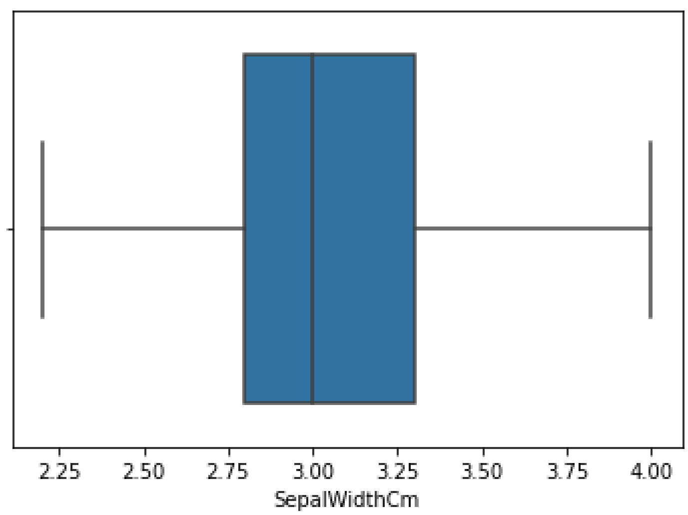
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f32698349d0>
```



```
# Importing
import sklearn
from sklearn.datasets import load_boston
import pandas as pd
import seaborn as sns

# Load the dataset
df = pd.read_csv('Iris dataset.csv')
```

```
# IQR
Q1 = np.percentile(df['SepalWidthCm'], 25,
                interpolation = 'midpoint')

Q3 = np.percentile(df['SepalWidthCm'], 75,
                interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df.shape)

# Upper bound
upper = np.where(df['SepalWidthCm'] >= (Q3+1.5*IQR))

# Lower bound
lower = np.where(df['SepalWidthCm'] <= (Q1-1.5*IQR))

# Removing the Outliers
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)

print("New Shape: ", df.shape)

sns.boxplot(x='SepalWidthCm', data=df)
```

```
Old Shape:  (150, 6)
New Shape:  (146, 6)
<matplotlib.axes._subplots.AxesSubplot at
0x7f326977d650>
```



REGRESSION MODEL

[ ] ↳ *5 cells hidden*

ACCURACY MODEL

[ ] ↳ *1 cell hidden*

CLUSTERING MODEL

[ ] ↳ *12 cells hidden*

PERORMING CLASSIFICATION

```
iris =  pd.read_csv("Iris dataset.csv")
```

```
from sklearn.model_selection import train_test_split

# Droping the target and species since we only need the measurements
X = iris.drop(['Id','Species'], axis=1)

# converting into numpy array and assigning petal length and petal width
X = X.to_numpy()[:, (2,3)]
y = iris['Id']

# Splitting into train and test
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.5, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
log_reg.fit(X,y)
```

```
LogisticRegression()
```

```
training_prediction = log_reg.predict(X_train)
training_prediction
```

```
array([ 77, 110,  60,  23,  99, 101,  23,  23,  63, 130,  14, 145,  23,
        23, 115,  99, 119, 135, 108, 119,  60,  23,  23, 135, 119,  23,
        23,  23, 135, 110,  23, 110, 119,  23, 135, 120, 115, 135, 145,
        14, 110, 107, 145,  80,  60,  74,  23,  74,  74,  23,  68, 119,
       119,  23,  99, 110, 123,  14, 110,  23, 135, 119, 123,  68, 123,
        74,  74, 115, 137,  14,  60, 115,  23,  68, 119])
```

```
test_prediction = log_reg.predict(X_test)
test_prediction
```

```
array([135,  23, 119,  77, 135,  23,  65, 145,  77,  63, 145,  14,  23,
        14,  23, 147, 119,  68,  74, 110,  14, 115,  23, 119, 119, 145,
       123, 119,  23,  14,  23,  23,  74,  14,  23, 115,  77,  23,  23,
        14, 145, 122, 134,  23,  23,  63, 135, 123,  74, 119,  91, 119,
        77,  23, 119,  68,  23,  23,  14,  80, 115,  13,  14,  23,  91,
        23,  68, 119,  23,  74, 115,  23, 115, 145,  68])
```

```
from sklearn import metrics

print("Precision, Recall, Confusion matrix, in training\n")

# Precision Recall scores
print(metrics.classification_report(y_train, training_prediction, digits=3))

# Confusion matrix
print(metrics.confusion_matrix(y_train, training_prediction))
```

```
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
```

```
print("Precision, Recall, Confusion matrix, in testing\n")

# Precision Recall scores
print(metrics.classification_report(y_test, test_prediction, digits=3))

# Confusion matrix
print(metrics.confusion_matrix(y_test, test_prediction))
```

```
  macro avg      0.017     0.049     0.021        75
weighted avg      0.019     0.053     0.023        75

[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 0 0]]
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Und
  _warn_prf(average, modifier, msg_start, len(result))
```

Conclusion: We have performed eda using different models where regression model have highest accuracy then a Other models