Department of
Computer Science and Engineering

Title: Statistical Analysis and Data Visualization using Python

Data Mining LAB

CSE 424

Green University of Bangladesh

1 Objective(s)

- To interpret data more easily and visualize the dataset effectively in data mining, machine learning and other data science tasks.

2 Data Visualization

Data visualization is defined as a graphical representation that contains the information and the data. By using visual elements like charts, graphs, and maps, data visualization techniques provide an accessible way to see and understand trends, outliers, and patterns in data.

Mainly, there are three different types of analysis for Data Visualization:

- **Univariate Analysis:** In the univariate analysis, we will be using a single feature to analyze almost all of its properties.

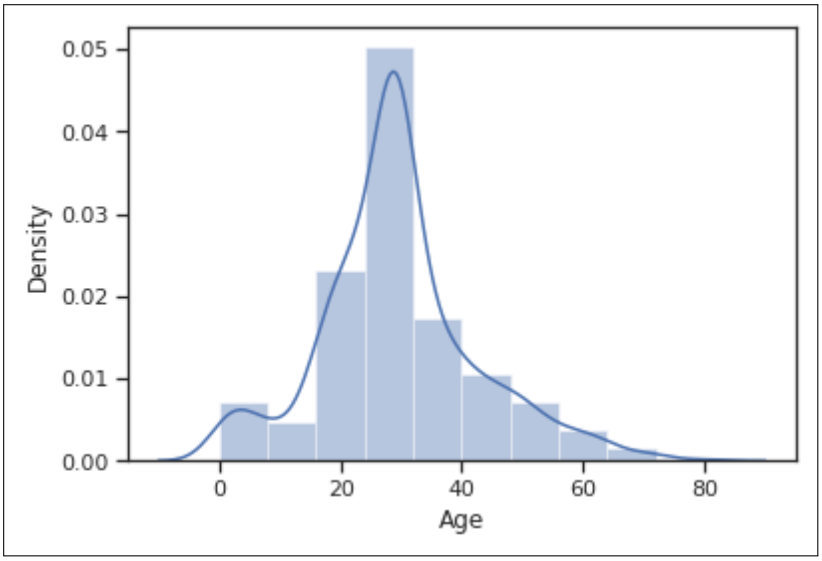
  1. **Distribution Plot:** When we want to analyze the impact on the target variable(output) with respect to an independent variable(input), we use distribution plots a lot. This plot gives us a combination of both probability density functions(pdf) and histogram in a single plot.

     Implementation in Python

```
# The distribution plot is present in the Seaborn package.

sns.distplot(updated_df['Age'], bins=10, kde=True, rug=False)
#kde is for kernel density estimate
```

     Input/Output

     Output of the programs is given below.



  2. **Box and Whisker Plot:** With the help of a box plot, we can determine the Interquartile range(IQR) where maximum details of the data will be present. Points that lie outside the whiskers (the straight lines at the maximum and minimum) will be considered as an outlier.

     Implementation in Python

```
sns.catplot(x="Categorical_Age", y="Age", kind="box", data=updated_df)
```

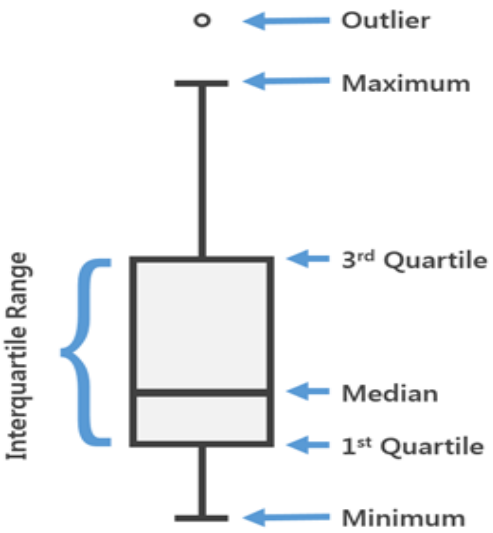
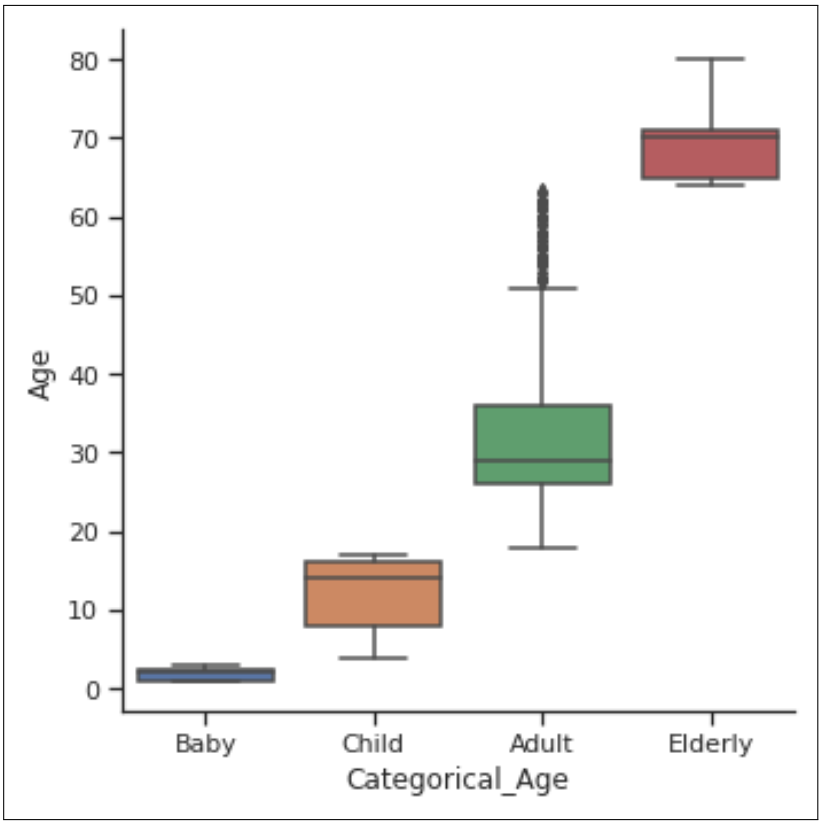


Figure 1: General Diagram for a Box-plot.

Input/Output

Output of the programs is given below.



3. **Violin Plot:** The violin plots can be considered as a combination of Box plot at the middle and distribution plots(Kernel Density Estimation) on both sides of the data. This can give us the description of the distribution of the dataset like whether the distribution is multimodal, Skewness, etc. It also gives us useful information like a 95% confidence interval.

Implementation in Python

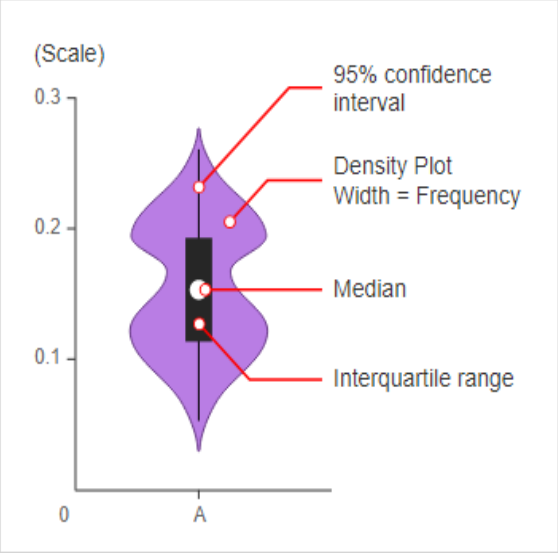


Figure 2: General Diagram for a Violin-plot.

```
# The Violin plot is present in the Seaborn package.

sns.violinplot(x='SurvStat',y='op_yr',data=hb,size=6)
```

- **Bivariate Analysis:** When we compare the data between exactly 2 features then it is known as bivariate analysis.

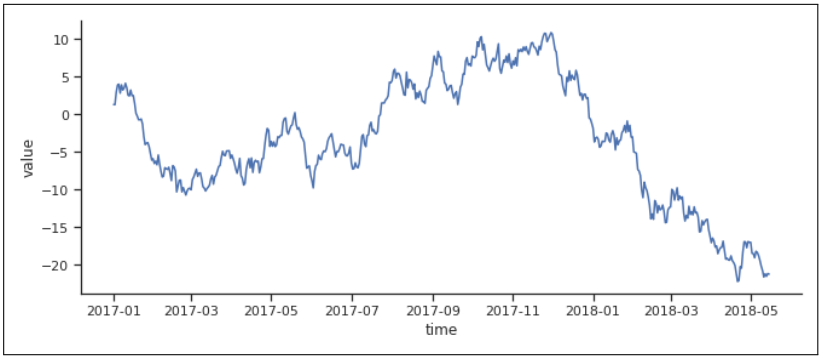
  1. **Line Plot:** The line plots are nothing but the values on a series of data points will be connected with straight lines.

     Implementation in Python

```
# The line plot is present in the Matplotlib package.

test_df = pd.DataFrame(dict(time=pd.date_range("2017-1-1", periods=500),
                            value=np.random.randn(500).cumsum()))
sns.relplot(x="time", y="value", kind="line", data=test_df ,height=4,
    aspect=15/6)
```

     Input/Output

     Output of the programs is given below.



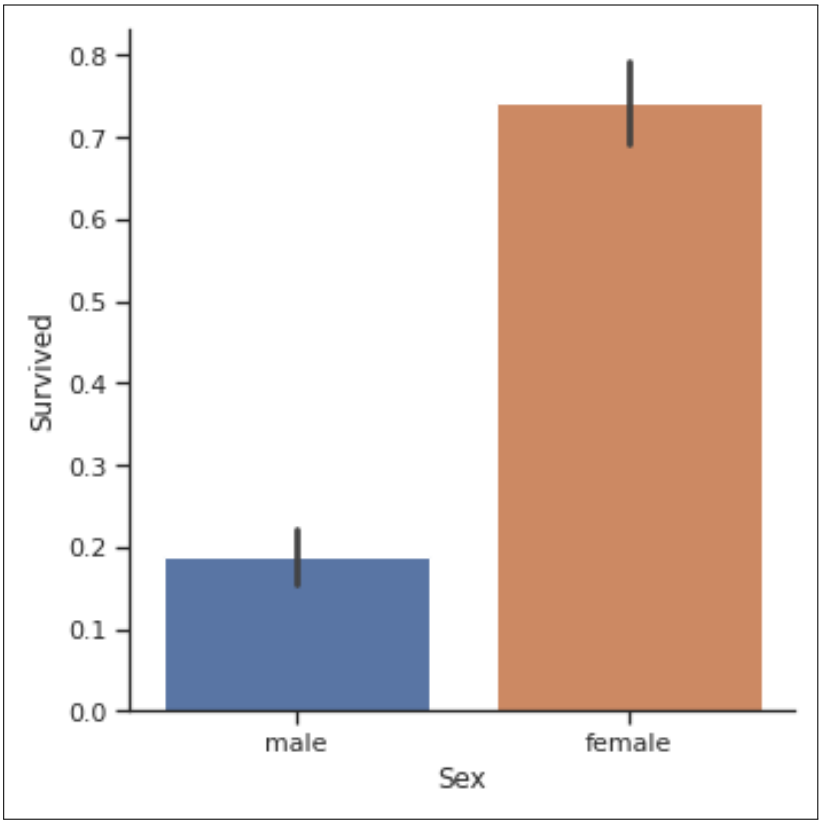
2. **Bar Plot:** Though it may seem simple it is powerful in analyzing data like sales figures every week, revenue from a product, Number of visitors to a site on each day of a week, etc.

Implementation in Python

```
# The bar plot is present in the Matplotlib package.

sns.catplot(x="Sex", y="Survived", kind="bar", data = updated_df)
```

Input/Output

Output of the programs is given below.



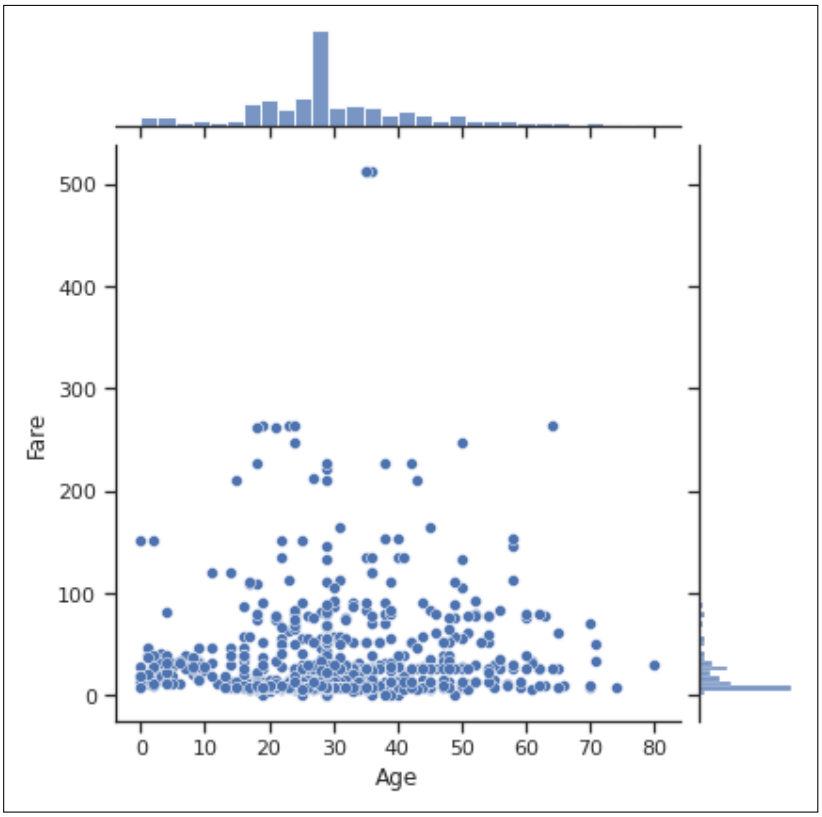
3. **Scatter Plot:** This plot describes us as a representation, where each point in the entire dataset is present with respect to any 2 to 3 features(Columns). Scatter plots are available in both 2-D as well as in 3-D. The 2-D scatter plot is the common one, where we will primarily try to find the patterns, clusters, and separability of the data.

Implementation in Python

```
# The scatter plot is present in the Matplotlib package.

sns.jointplot(x="Age", y="Fare", data=updated_df)
```

Input/Output

Output of the programs is given below.



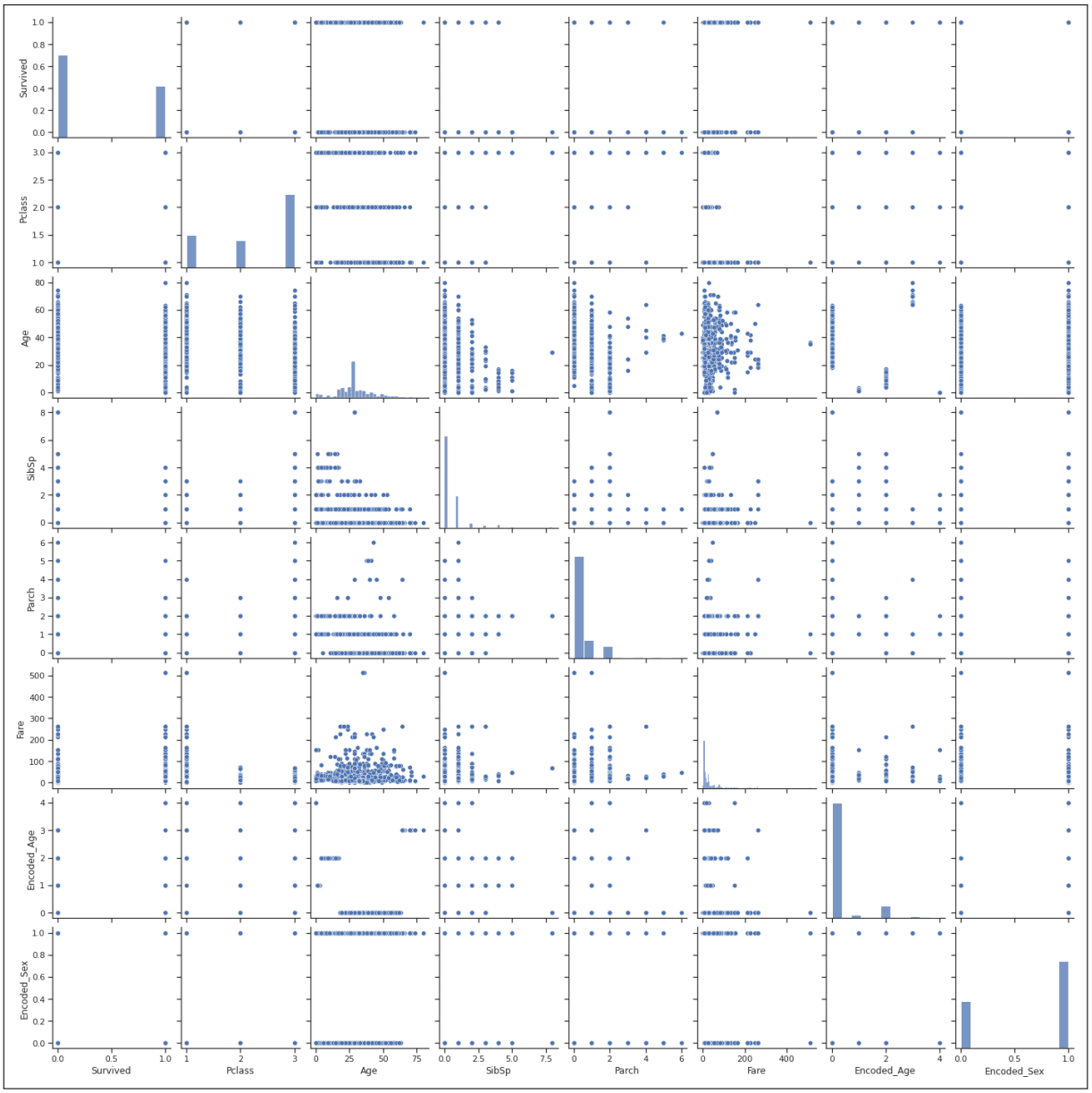
Using seaborn library it is possible to show the relationship among every pair of features as follows:

Implementation in Python

```
# The scatter plot is present in the Matplotlib package.

sns.pairplot(updated_df)
```

Input/Output

Output of the programs is given below.

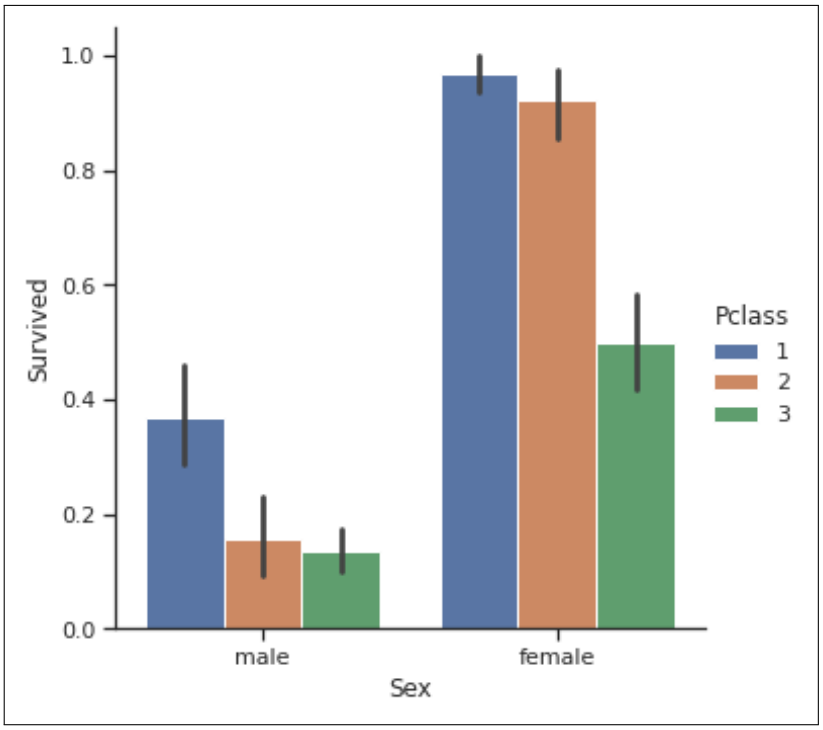


- **Multivariate Analysis:** In the multivariate analysis, we will be comparing more than 2 variables.

```
# The bar plot is present in the Matplotlib package.

sns.catplot(x="Sex", y="Survived", hue="Pclass", kind="bar", data =
    updated_df)
```

Input/Output

Output of the programs is given below.



3 Feature selection

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.
We can summarize feature selection as follows (Figure 1).

- **Feature Selection:** Select a subset of input features from the dataset.
  - **Unsupervised:** Do not use the target variable (e.g. remove redundant variables).
    - Correlation

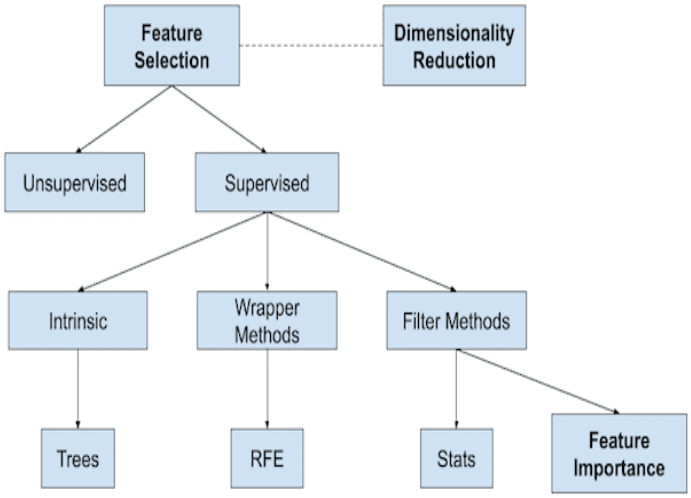


Figure 3: Overview of Feature Selection Techniques.

  - **Supervised:** Use the target variable (e.g. remove irrelevant variables).
    - **Wrapper:** Search for well-performing subsets of features.

- RFE
* **Filter:** Select subsets of features based on their relationship with the target.
  - Statistical Methods
  - Feature Importance Methods
* **Intrinsic:** Algorithms that perform automatic feature selection during training.
  - Decision Trees

3.1 Pearson's Correlation

The Pearson (product-moment) correlation coefficient is a measure of the linear relationship between two features. It's the ratio of the covariance of x and y to the product of their standard deviations. It's often denoted with the letter r and called Pearson's r. You can express this value mathematically with this equation:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Here are some important facts about the Pearson correlation coefficient:

- The Pearson correlation coefficient can take on any real value in the range $-1 \leq r \leq 1$.
- The maximum value $r = 1$ corresponds to the case in which there's a perfect positive linear relationship between x and y. In other words, larger x values correspond to larger y values and vice versa.
- The value $r > 0$ indicates positive correlation between x and y.
- The value $r = 0$ corresponds to the case in which there's no linear relationship between x and y.
- The value $r < 0$ indicates negative correlation between x and y.
- The minimal value $r = -1$ corresponds to the case when there's a perfect negative linear relationship between x and y. In other words, larger x values correspond to smaller y values and vice versa.

Implementation in Python

```
# The Pearson's Correlation is present in the Matplotlib package.

print(updated_df["Age"].corr(updated_df["Survived"]))
print(updated_df["Age"].corr(updated_df["Survived"], method='spearman')) #
    Spearman's rho
```

Input/Output

Output of the programs is given below.

```
-0.0678091474877425
-0.03702099645621949
```

3.2 Feature Importance using Random Forest

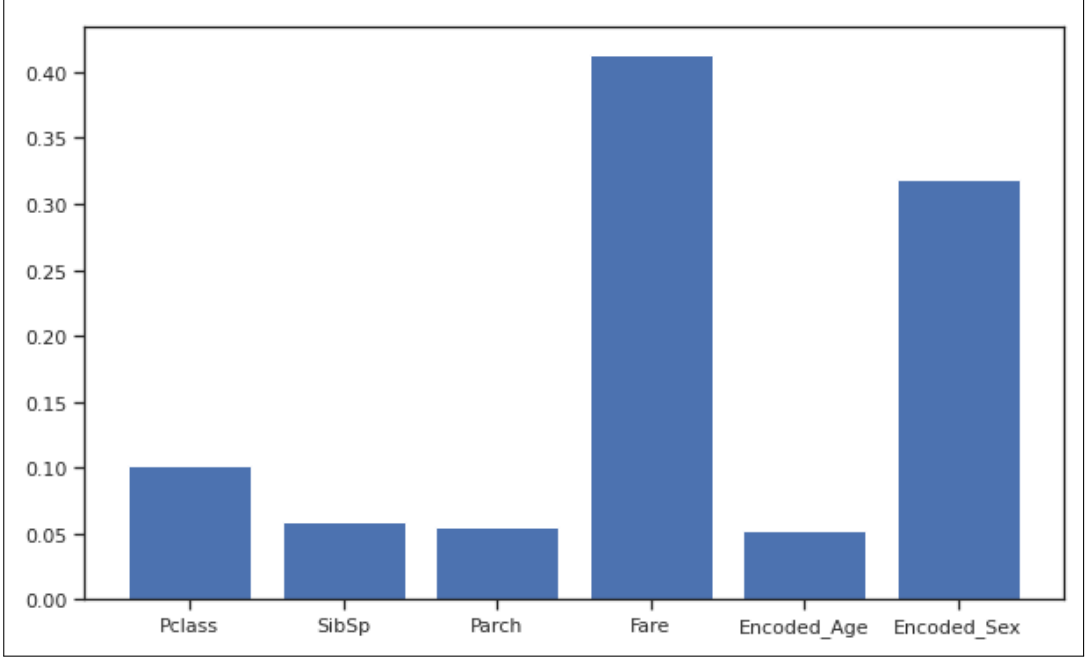
```
# random forest for feature importance on a classification problem
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from matplotlib import pyplot
plt_1 = plt.figure(figsize=(10, 6))

# define dataset
X = updated_df[['Pclass','SibSp','Parch','Fare','Encoded_Age','Encoded_Sex']]
col = ['Pclass','SibSp','Parch','Fare','Encoded_Age','Encoded_Sex']
```

```
print(X)
y = updated_df["Survived"]
# define the model
model = RandomForestClassifier()
# fit the model
model.fit(X, y)
# get importance
importance = model.feature_importances_
# summarize feature importance
for i,v in enumerate(importance):
  print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
pyplot.bar([x for x in col], importance)
pyplot.show()
```

Input/Output

Output of the programs is given below.

```
Feature: 0, Score: 0.10128
Feature: 1, Score: 0.05887
Feature: 2, Score: 0.05472
Feature: 3, Score: 0.41342
Feature: 4, Score: 0.05276
Feature: 5, Score: 0.31896
```



4 Discussion & Conclusion

Based on the focused objective(s) to understand about the python program to visualize and analysis data, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

5 Lab Exercise (Submit as a report)

- Develop a jupyter notebook in Colab or Kaggle using all of the data processing criteria shown in the class. Choose an appropriate dataset from kaggle or any other sources containg NULL and garbage values (If

not found, manipulate dataset by your own), then show the effect of all visualizing techniques with proper documentation in the notebook.

- Create two dataframe and show their correlation using Pearson's correlation and visualize with appropriate plot.

6 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected