Name – Gurucharan Rajendra Kapale
Roll No. 22
Class – TE
Title – Prac 8 : Data Visualization I

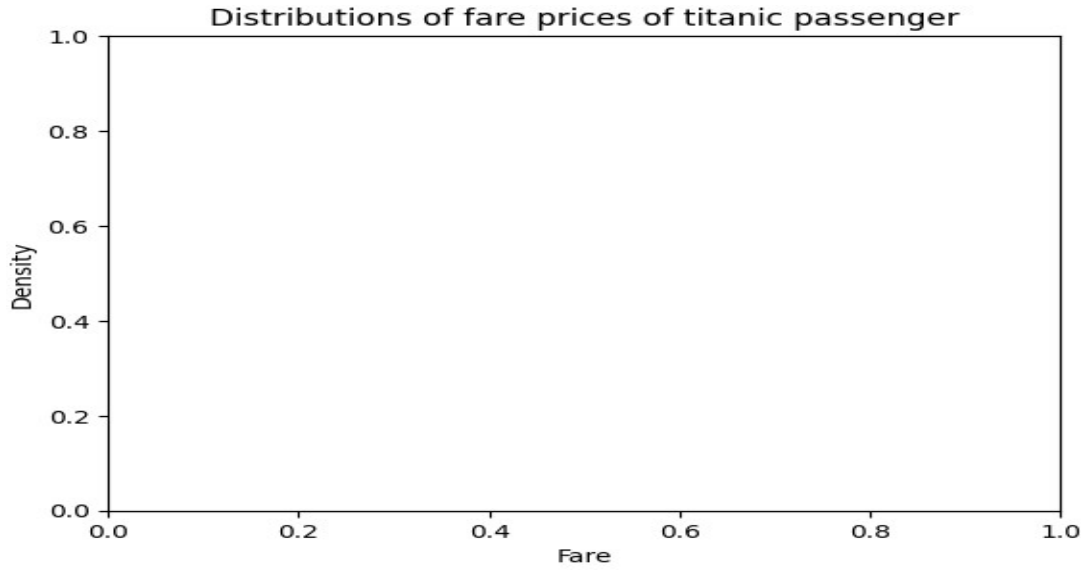
1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.

2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

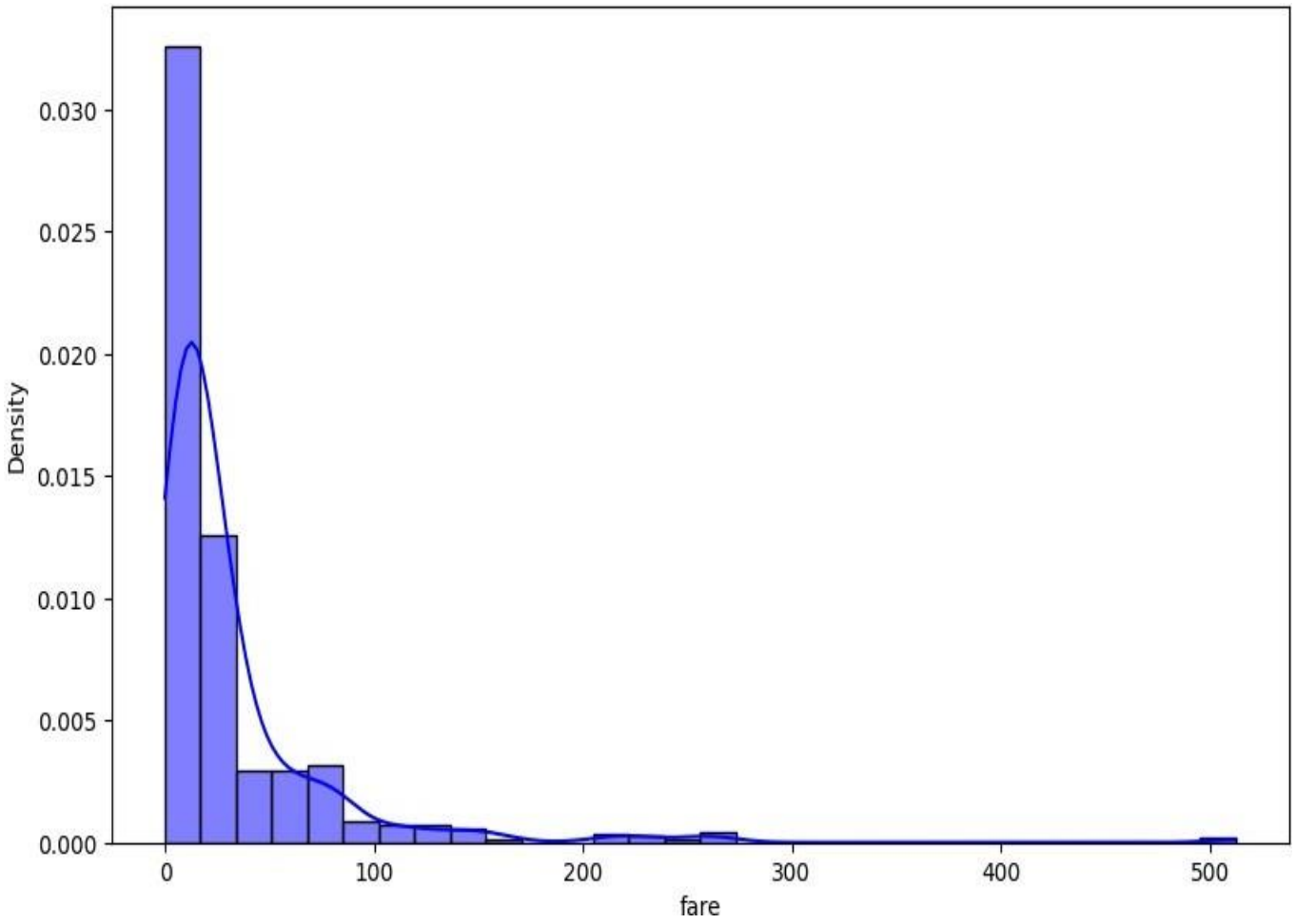
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

titanic=sns.load_dataset('titanic')
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton

```
plt.title('Distributions of fare prices of titanic passenger')
plt.xlabel('Fare')
plt.ylabel('Density')
plt.show()
```



```
plt.figure(figsize=(10,6))
sns.histplot(titanic['fare'],bins=30, kde=True , color='blue' ,stat='density' )
```



Name – Gurucharan Rajendra Kapale
Roll No. 22
Class – TE
Title – Prac 9 : Data Visualization II
1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age')

2. Write observations on the inference from the above statistics.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

titanic = sns.load_dataset('titanic')
titanic.head(10)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampto
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampto
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampto
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampto
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	NaN	Queenstown
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampto
7	0	3	male	2.0	3	1	21.0750	S	Third	child	False	NaN	Southampto
8	1	3	female	27.0	0	2	11.1333	S	Third	woman	False	NaN	Southampto
9	1	2	female	14.0	1	0	30.0708	C	Second	child	False	NaN	Cherbourg

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
```

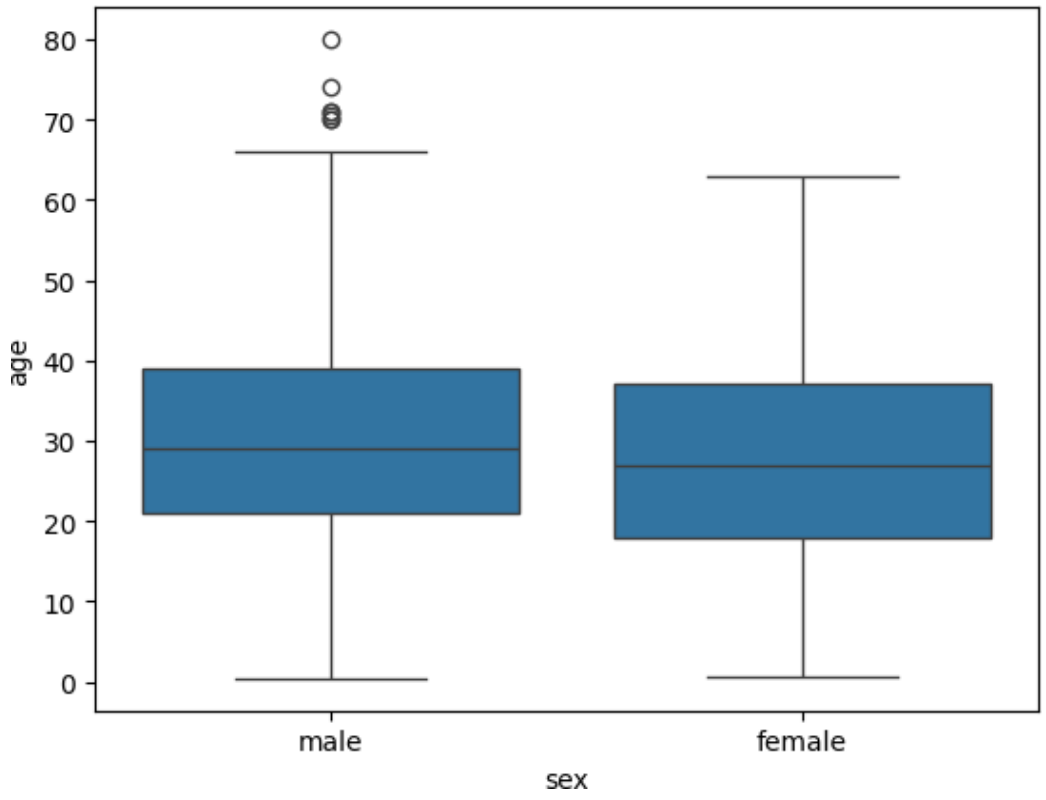
```
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

titanic.loc[:,["survived","alive"]]

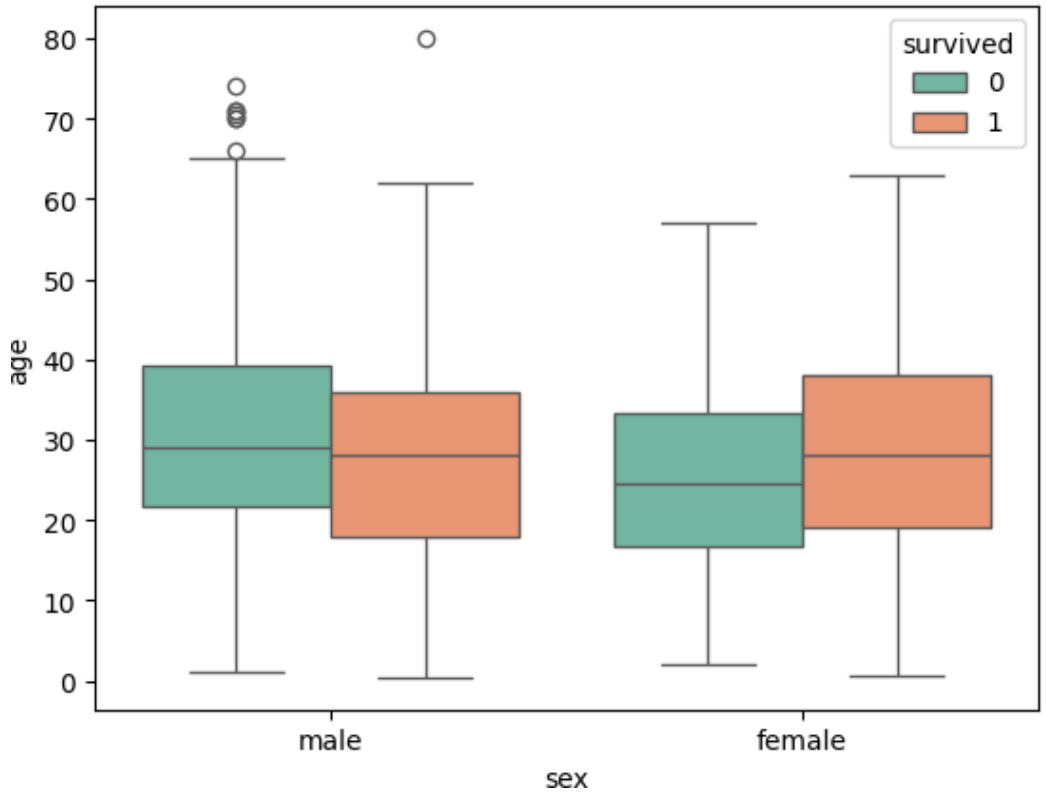
	survived	alive
0	0	no
1	1	yes
2	1	yes
3	1	yes
4	0	no
...	...	...
886	0	no
887	1	yes
888	0	no
889	1	yes
890	0	no

891 rows × 2 columns

sns.boxplot(x='sex', y='age', data=titanic)
plt.show



sns.boxplot(x='sex', y='age', hue='survived', data=titanic, palette="Set2")



Name – Gurucharan Rajendra Kapale
Roll No. 22
Class – TE
Title – Prac 10 : Data Visualization III
Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as:

1. List down the features and their types (e.g., numeric, nominal) available in the dataset.
2. Create a histogram for each feature in the dataset to illustrate the feature distributions.
3. Create a boxplot for each feature in the dataset.
4. Compare distributions and identify outliers

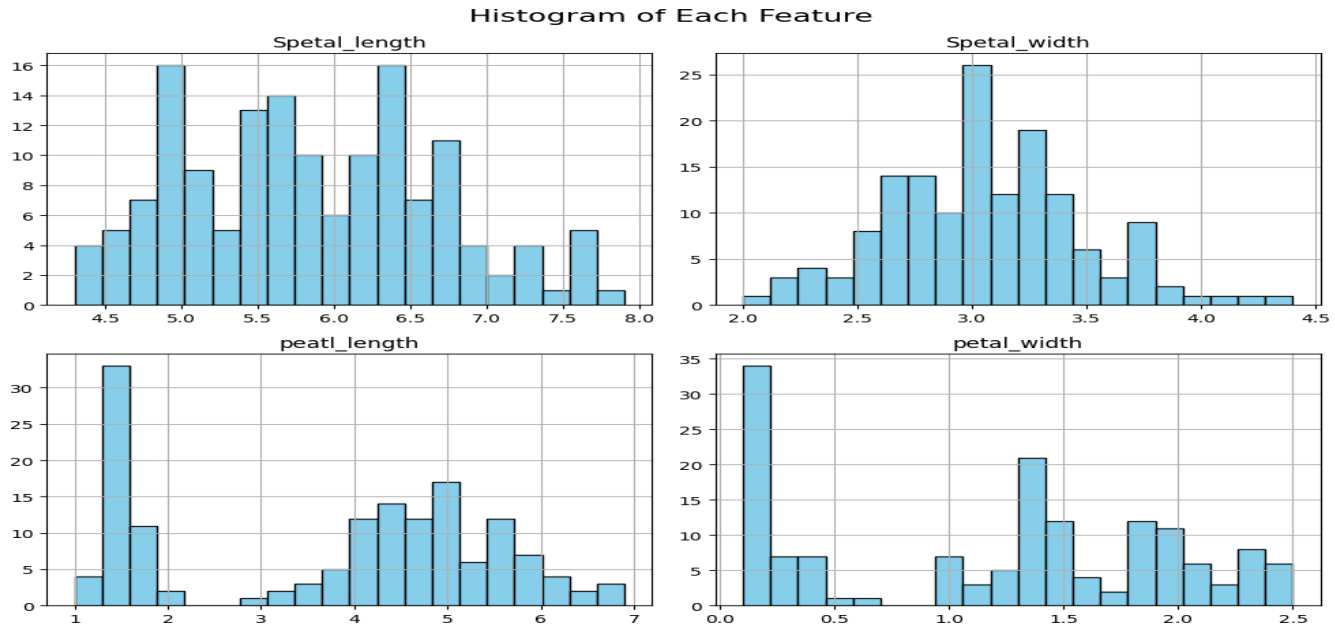
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('iris.csv')

print("Features And Types in the iris dataset")
```

```
Features And Types in the iris dataset
```

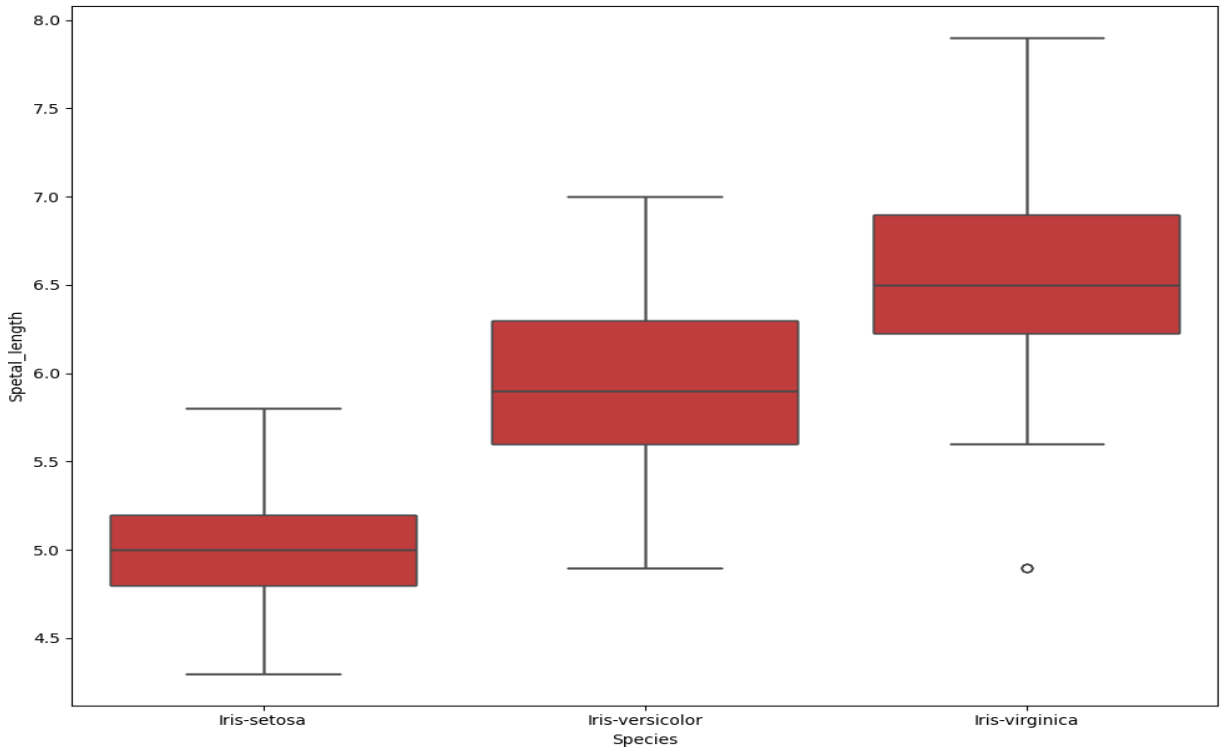
```
print (df.dtypes)
```

```
Spetal_length     float64
Spetal_width      float64
peatl_length      float64
petal_width       float64
Species            object
dtype: object
```

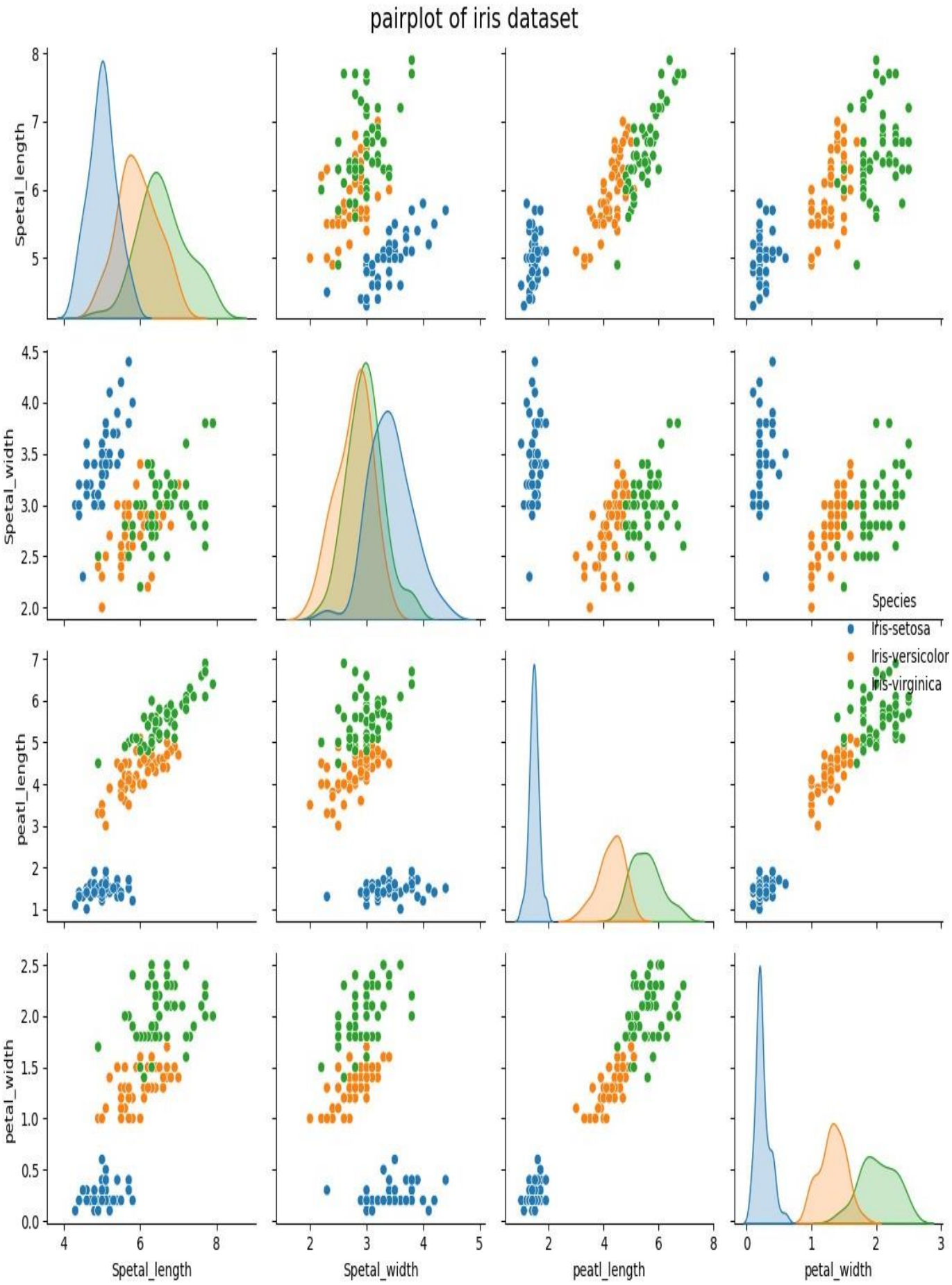
```
df.drop(columns='Species').hist(bins=20, figsize=(10,8), color='skyblue',edgecolor='black')
plt.suptitle('Histogram of Each Feature',fontsize=16)
plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(10,8))
sns.boxplot(x='Species',y='Spetal_length',data=df)
sns.boxplot(x='Species',y='Spetal_length',data=df)
sns.boxplot(x='Species',y='Spetal_length',data=df)
sns.boxplot(x='Species',y='Spetal_length',data=df)
plt.tight_layout()
plt.show()
```




```
sns.pairplot(df,hue='Species')
plt.suptitle('pairplot of iris dataset',fontsize=16)
plt.tight_layout()
plt.show()
```


pairplot of iris dataset
Spetal_length
Spetal_width
peatl_length
petal_width
Species
Iris-setosa
Iris-versicolor
Iris-virginica

Name – Gurucharan Rajendra Kapale
Roll No. 22
Class – TE
Title – Prac 7 : Text Analytics
1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.

2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk import pos_tag
from sklearn.feature_extraction.text import TfidfVectorizer
import string

nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger_eng')

document ="""
Natural Langauage processing is a field  of AI  It aims to enable to understand ,interpret and generate human langaueg
"""

tokens= word_tokenize(document)
pos_tags = pos_tag(tokens)
stop_words = set(stopwords.words('english'))
filtered_tokens=[word for word in tokens if word.lower() not in stop_words and word not in string.punctuation]
stemmer =PorterStemmer()
stemmed_tokens=[stemmer.stem(word) for word in filtered_tokens]
lemmatizer = WordNetLemmatizer()
lemmatized_tokens =[lemmatizer.lemmatize(word)for word in filtered_tokens]

print("\n original doc\n",document)
print("\n tokens\n", tokens)
print("\n pos tag\n",pos_tag)
print("\n filtered token after removed stop word removal\n", filtered_tokens)
print("\n stemmed tokens\n",stemmed_tokens)
print("\n Lemmetaized Tokens\n",lemmatized_tokens)
```

```
 original doc

Natural Langauage processing is a field  of AI  It aims to enable to
understand ,interpret and generate human langaueg


 tokens
 ['Natural', 'Langauage', 'processing', 'is', 'a', 'field', 'of', 'AI', 'It',
'aims', 'to', 'enable', 'to', 'understand', ',', 'interpret', 'and', 'generate',
'human', 'langaueg']

 pos tag
 <function pos_tag at 0x7fa62b0a5700>

 filtered token after removed stop word removal
 ['Natural', 'Langauage', 'processing', 'field', 'AI', 'aims', 'enable',
'understand', 'interpret', 'generate', 'human', 'langaueg']

 stemmed tokens
 ['natur', 'langauag', 'process', 'field', 'ai', 'aim', 'enabl', 'understand',
'interpret', 'gener', 'human', 'langaueg']

 Lemmetaized Tokens
 ['Natural', 'Langauage', 'processing', 'field', 'AI', 'aim', 'enable',
'understand', 'interpret', 'generate', 'human', 'langaueg']



corpus=[document]

vectorizer = TfidfVectorizer()

x=vectorizer.fit_transform(corpus)
```

tfidf_matrix= x.toarray()
terms= vectorizer.get_feature_names_out()

print("\nTerm Frequency And Inverse Document Frequency\n")
for i,term in enumerate(terms):
   print(f"Term:(term),TF-IDF:(tfidf_matrix[0][i])")

```
Term Frequency And Inverse Document Frequency

Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
Term:(term),TF-IDF:(tfidf_matrix[0][i])
```