List of Practical for the viva.The practical exam will be conducted on the given practical list for ADS.
NOTE: The DATASET will be provided to you on the same day. You will not be using any dataset form google.
A link will be shared which will have all the datasets.
For any doubts msg me in personal.

1) Explore the descriptive
(mean, median, minimum, maximum, standard deviation) and inferential statistics
on the Olympic dataset.

```
[5]: statistics_by_gender = data.groupby('gender').describe()
```

```
[6]: statistics_by_gender
```

	Unnamed: 0								age		...	health_wellness		miscellaneous							
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%	50%	75%	max
gender																					
Female	323.0	469.331269	299.491513	1.0	203.5	433.0	730.0	996.0	323.0	21.597523	...	156.50	200.0	323.0	108.671827	51.325451	20.0	67.0	108.0	149.50	200.0
Male	356.0	507.542135	285.791662	10.0	256.0	517.5	745.5	997.0	356.0	21.668539	...	159.25	200.0	356.0	107.030899	52.578222	21.0	60.0	106.5	153.25	200.0
Non-binary	321.0	520.937695	279.461473	0.0	288.0	535.0	760.0	999.0	321.0	21.760125	...	157.00	200.0	321.0	111.233645	53.375248	20.0	65.0	117.0	157.00	200.0

3 rows × 112 columns

2) Use SMOTE technique to
generate synthetic data on diabetic dataset.

```
from imblearn.over_sampling import SMOTE
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import numpy as np

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
print(np.bincount(y_train))
print(np.bincount(y_resampled))
```

3) Outlier detection using distance-based method on Olympic dataset.

```
import pandas as pd

import numpy as np

from sklearn.neighbors import NearestNeighbors


data = pd.read_csv('olympic_data.csv')

numeric_data = data.select_dtypes(include=[np.number])

numeric_data = numeric_data.dropna()

k = 5

knn_model = NearestNeighbors(n_neighbors=k)

knn_model.fit(numeric_data)

distances, indices = knn_model.kneighbors()

avg_distances = np.mean(distances, axis=1)

threshold = np.mean(avg_distances) + 2 * np.std(avg_distances)

outliers_indices = np.where(avg_distances > threshold)[0]

outliers = numeric_data.iloc[outliers_indices]

print("Outliers:")

print(outliers)
```

4) Implement time series
forecasting on international-airline-passengers.csv.

```
import pandas as pd

from statsmodels.tsa.arima.model import ARIMA

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split


url = 'https://raw.githubusercontent.com/ejgao/Time-Series-Datasets/master/Electric_Production.csv'

df = pd.read_csv(url, header=0, parse_dates=[0], index_col=0)


train, test = train_test_split(df, test_size=0.2, shuffle=False)


order = (5, 1, 0)


model = ARIMA(train, order=order)

model_fit = model.fit()


predictions = model_fit.forecast(steps=len(test))


plt.figure(figsize=(12, 6))

plt.plot(train, label='Training Data')

plt.plot(test, label='Actual Data')

plt.plot(test.index, predictions, label='Predictions', color='red')

plt.title('Time Series Forecasting with ARIMA')

plt.legend()

plt.show()
```

5) Illustrate data science lifecycle for any of the dataset.

theory

6) Implement and explore
performance evaluation metrics for housing dataset.

```
import pandas as pd

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

import numpy as np


data = pd.read_csv('housing_data.csv')


actual_values = data['actual_values'].values

predicted_values = data['predicted_values'].values


mae = mean_absolute_error(actual_values, predicted_values)

print("Mean Absolute Error (MAE):", mae)


mse = mean_squared_error(actual_values, predicted_values)

print("Mean Squared Error (MSE):", mse)


rmse = np.sqrt(mse)

print("Root Mean Squared Error (RMSE):", rmse)


r_squared = r2_score(actual_values, predicted_values)

print("R-squared (R²):", r_squared)


def mean_absolute_percentage_error(y_true, y_pred):

   y_true, y_pred = np.array(y_true), np.array(y_pred)

   return np.mean(np.abs((y_true - y_pred) / y_true)) * 100


mape = mean_absolute_percentage_error(actual_values, predicted_values)

print("Mean Absolute Percentage Error (MAPE):", mape)
```

7) Implement and explore
performance evaluation metrics for placement dataset.

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, roc_curve, roc_auc_score

import matplotlib.pyplot as plt


data = pd.read_csv('placement_data.csv')

X = data.drop('placement_status', axis=1)

y = data['placement_status']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


model = RandomForestClassifier()

model.fit(X_train, y_train)


y_pred = model.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)

precision = precision_score(y_test, y_pred)

recall = recall_score(y_test, y_pred)

f1 = f1_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)


print("Accuracy:", accuracy)

print("Precision:", precision)

print("Recall:", recall)

print("F1 Score:", f1)

print("Confusion Matrix:")

print(conf_matrix)


y_pred_proba = model.predict_proba(X_test)[:,1]
```

```
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
auc = roc_auc_score(y_test, y_pred_proba)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

8) Perform data Imputation
on Automobile dataset.

https://chat.openai.com/share/08e5b06c-0cbf-44c5-bc26-d898567e69b7

9) Explore data
visualization techniques on placement dataset.

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

tips = sns.load_dataset("tips")

sns.scatterplot(x="total_bill", y="tip", data=tips)
plt.title("Total Bill vs Tip")
plt.xlabel("Total Bill ($)")
plt.ylabel("Tip ($)")
plt.show()

sns.countplot(x="day", data=tips)
plt.title("Count of Observations by Day")
plt.xlabel("Day of the Week")
plt.ylabel("Count")
plt.show()

sns.histplot(tips["total_bill"], bins=15, kde=True)
plt.title("Distribution of Total Bill Amount")
plt.xlabel("Total Bill ($)")
plt.ylabel("Frequency")
plt.show()

plt.figure(figsize=(8, 8))
tip_sizes = tips["sex"].value_counts()
plt.pie(tip_sizes, labels=tip_sizes.index, autopct='%1.1f%%',
startangle=140)
plt.title("Proportion of Tips by Gender")
plt.axis('equal')
plt.show()

corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```

10) Using Box blot find out
the outliers for any of the dataset given in the folder.

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


data = pd.read_csv('your_dataset.csv')


sns.boxplot(data=data)


plt.show()
```

11) Write a python program
to find inferential statistics.

```
import pandas as pd

from scipy.stats import chi2_contingency


data=pd.read_csv('athlete_events.csv')

contingency_table=pd.crosstab(data['Age'],data['Height'])

chi2_stat,p_value,dof,excepted = chi2_contingency(contingency_table)

print(f"Chi-stats:{chi2_stat}")

print(f"p_value:{p_value}")

print(f"Degree of freedom :{dof}")

print(f"excepted frequency:{excepted}")
```

1. **T-test:**
   - **Independent Samples T-test:** Used to compare means of two independent groups.
     - Formula for the t-statistic: $t = \frac{\bar{X}_1 - \bar{X}_2}{s_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}$
       - $\bar{X}_1$ and $\bar{X}_2$: sample means of the two groups
       - s_p: pooled standard deviation
       - n_1 and n_2: sample sizes of the two groups
   - **One-sample T-test:** Used to test the mean of a single group against a known mean.
     - Formula for the t-statistic: $t = \frac{\bar{X} - \mu}{s/\sqrt{n}}$
       - $\bar{X}$: sample mean
       - μ: population mean
       - s: sample standard deviation
       - n: sample size

- n: sample size

2. **Z-test**:
   - Used to compare means of two independent groups when the population standard deviation is known or the sample size is large (usually greater than 30).
   - Formula for the z-score: $z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$
     - $\bar{X}_1$ and $\bar{X}_2$: sample means of the two groups
     - σ_1 and σ_2: population standard deviations of the two groups
     - n_1 and n_2: sample sizes of the two groups
3. **Chi-square test**:
   - Used to test the association between two categorical variables.
   - Formula for the chi-square statistic depends on the type of chi-square test (e.g., chi-square test for independence, chi-square goodness of fit).
   - **Chi-square test for independence**:
     - Formula for the chi-square statistic: $\chi^2 = \sum \frac{(O-E)^2}{E}$
       - O: Observed frequency
       - E: Expected frequency

12) Perform Exploratory Data
Analysis (EDA) on Automobile csv

handle missing data

Handle categorical data

Standard scalar

```
import pandas as pd

data = pd.read_csv("Social_Network_Ads.csv")
X = data.iloc[:,2:4].values
Y = data.iloc[:,4:5].values

from sklearn.impute import SimpleImputer
si = SimpleImputer()
X = si.fit_transform(X)

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
Y = le.fit_transform(Y)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)

from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(min_samples_split=3)
dtc.fit(X, Y)

newData = pd.read_csv("newinformation.csv")
newX = newData.iloc[:,2:4].values
pred = dtc.predict(newX)
print(le.inverse_transform(pred))
```

14) Perform Visualize
correlation between sepal length and petal length in iris data set using scatter plot.

13) Explore data
visualization techniques like scatter plot and show correlation.

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()

iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

iris_df['species'] = iris.target

iris_df['species'] = iris_df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

sns.scatterplot(data=iris_df, x='sepal length (cm)', y='petal length (cm)', hue='species')
plt.title('Correlation between Sepal Length and Petal Length')
plt.show()
```

15) Perform univariate analysis
like Mean, median, variance, Standard deviation, skewness, and kurtosis on Diabetes dataset.

```
import numpy as np

import pandas as pd

from sklearn.datasets import load_diabetes

from scipy.stats import skew, kurtosis


diabetes = load_diabetes()

data = pd.DataFrame(data=diabetes.data, columns=diabetes.feature_names)


statistics = pd.DataFrame(index=data.columns)

statistics['Mean'] = data.mean()

statistics['Median'] = data.median()

statistics['Variance'] = data.var()

statistics['Standard Deviation'] = data.std()

statistics['Skewness'] = data.apply(skew)

statistics['Kurtosis'] = data.apply(kurtosis)


print(statistics)
```