DATA SCIENCE AND BIG DATA SCIENCE PRACTICALS

Assignment 1

```
import pandas as pd

df = pd.read_csv("revenue.csv")
df.head(10)

df.plot(x='company', y='revenue', kind='bar')

df.plot(x='company', y='revenue',kind='bar', logy=True)


untitled ipyb
import pandas as pd

df = pd.read_csv("revenue.csv")
df.head(10)

df.plot(x='company', y='revenue', kind='pie')
```

Assignment 2

outlier remover using z score

```
import pandas as pd
import matplotlib
from matplotlib import pyplot as plt
%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (12,8)

df = pd.read_csv("bhp.csv")
df.head()

df.price_per_sqft.describe()

plt.hist(df.price_per_sqft, bins=20, rwidth=0.8)
plt.xlabel('Price per square ft')
plt.ylabel('Count')
plt.show()

plt.hist(df.price_per_sqft, bins=20, rwidth=0.8)
plt.xlabel('Price per square ft')
plt.ylabel('Count')
plt.yscale('log')
plt.show()
```

Treat outliers using percentile first

```
lower_limit, upper_limit = df.price_per_sqft.quantile([0.001, 0.999])
lower_limit, upper_limit

outliers = df[(df.price_per_sqft>upper_limit) | (df.price_per_sqft<lower_limit)]
outliers.sample(10)

df2 = df[(df.price_per_sqft<upper_limit) & (df.price_per_sqft>lower_limit)]
df2.shape

df.shape

df.shape[0] – df2.shape[0]
```

NOW REMOVE OUTLIERS USING 4 STANDARD DEVIATION

```
max_limit = df2.price_per_sqft.mean() + 4*df2.price_per_sqft.std()
min_limit = df2.price_per_sqft.mean() - 4*df2.price_per_sqft.std()
max_limit, min_limit

df2[(df2.price_per_sqft>max_limit) | (df2.price_per_sqft<min_limit)].sample(10)

df3 = df2[(df2.price_per_sqft>min_limit) & (df2.price_per_sqft<max_limit)]
df3.shape

df2.shape[0]-df3.shape[0]

plt.hist(df3.price_per_sqft, bins=20, rwidth=0.8)
plt.xlabel('Price per square ft')
plt.ylabel('Count')
plt.show()

from scipy.stats import norm
import numpy as np

plt.hist(df3.price_per_sqft, bins=20, rwidth=0.8, density=True)
plt.xlabel('Height (inches)')
plt.ylabel('Count')

rng = np.arange(-5000, df3.price_per_sqft.max(), 100)
plt.plot(rng, norm.pdf(rng,df3.price_per_sqft.mean(),df3.price_per_sqft.std()))
```

NOW REMOVE OUTLIERS USING TRESOLD

```
df2['zscore'] = (df2.price_per_sqft-df2.price_per_sqft.mean())/df2.price_per_sqft.std()
df2.sample(10)

outliers_z = df2[(df2.zscore < -4) | (df2.zscore>4)]
outliers_z.shape
```

```
outliers_z.sample(5)

df4 = df2[(df2.zscore>-4)&(df2.zscore<4)]
df4.shape
df2.shape[0] – df4.shape[0]
```

Normal distribution (assignment 2)

```
import pandas as pd
import seaborn as sn

df = pd.read_csv("heights.csv")
df.head()
```

Outlier detection and removal using standard deviation

```
df.height.describe()

sn.histplot(df.height, kde=True)

mean = df.height.mean()
mean

std_deviation = df.height.std()
std_deviation

mean-3*std_deviation

mean+3*std_deviation

df[(df.height < 54.82) | (df.height > 77.91)]

df_no_outlier = df[(df.height<77.91) & (df.height>54.82)]
df_no_outlier.shape

df['zscore'] = ( df.height - df.height.mean() ) / df.height.std()
df.head(5)

df.height.mean()

df.height.std()
df[df['zscore']>3]

df[df['zscore']<-3]
```

Assignment 3

```
import pandas as pd

df = pd.read_csv("AB_NYC_2019.csv")
```

```
df.head()

df.price.describe()

min_thresold, max_thresold = df.price.quantile([0.01,0.999])
min_thresold, max_thresold

df[df.price<min_thresold]

df2 = df[(df.price>min_thresold)&(df.price<max_thresold)]
df2.shape

df2.sample(5)

df2.price.describe()


import pandas as pd
import numpy as np

df = pd.read_csv("income.csv", names=["name","income"], skiprows=[0])
df

help(pd.read_csv)

df.income.describe()

df.income.quantile(0)

df.income.quantile(0.25,interpolation="higher")

df.income.quantile(0.5,interpolation="higher")

df.income.quantile(0.75)

percentile_99 = df.income.quantile(0.99)
percentile_99

df[df.income>percentile_99]

df['income'][3]=np.NaN

df

df.income.mean()

df_new = df.fillna(df.income.mean())
df_new

df_new = df.fillna(df.income.median())
df_new
```

Assignment No-4

```
import pandas as pd
import seaborn as sns
import numpy as np

df = pd.read_csv(
    "income.csv",
    index_col=None,
    names=["income","count"],
    skiprows=1
)
df.head()

sns.set(rc={'figure.figsize':(11.7,8.27)})
g = sns.barplot(x='income',y='count',data=df)
g.set_xticklabels(g.get_xticklabels(),
                  rotation=45,
                  horizontalalignment='left');

sns.set(rc={'figure.figsize':(11.7,8.27)})
g = sns.barplot(x='income',y='count',data=df)
g.set_xticklabels(g.get_xticklabels(),
                  rotation=45,
                  horizontalalignment='right');
g.set(xscale="log");

plt.bar(x=df["income"],y=df["count"])
```

Assignment no.5

```
import pandas as pd
df = pd.read_csv("carprices.csv")
df

dummies = pd.get_dummies(df['Car Model'])
dummies

merged = pd.concat([df,dummies],axis='columns')
merged

final = merged.drop(["Car Model","Mercedez Benz C class"],axis='columns')
final

y = final['Sell Price($)']
y

from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

```
model.fit(X,y)

model.score(X,y)

model.predict([[45000,4,0,0]])

model.predict([[86000,7,0,1]])
```

ASSIGNMENT 5

CATEGORIAL VARIABLES AND ONE HOT CODING

```
import pandas as pd

df = pd.read_csv("homeprices.csv")
df
```

USING PANDAS TO CREATE DUMMY VARIABLES

```
dummies = pd.get_dummies(df.town)
dummies

merged = pd.concat([df,dummies],axis='columns')
merged

final = merged.drop(['town'], axis='columns')
final
```

DUMMY VARIABLES TRAP

```
final = final.drop(['west windsor'], axis='columns')
final

X = final.drop('price', axis='columns')
X

y = final.price

from sklearn.linear_model import LinearRegression
model = LinearRegression()

model.fit(X,y)

model.predict(X) # 2600 sqr ft home in new jersey

model.score(X,y)

model.predict([[3400,0,0]]) # 3400 sqr ft home in west windsor

model.predict([[2800,0,1]]) # 2800 sqr ft home in robbinsville
```

USING SKLEARN ONEHOTENCODER

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dfle = df
dfle.town = le.fit_transform(dfle.town)
dfle

X = dfle[['town','area']].values
X

y = dfle.price.values
y

from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
ct = ColumnTransformer([('town', OneHotEncoder(), [0])], remainder = 'passthrough')

X = ct.fit_transform(X)
X

X = X[:,1:]
X

model.fit(X,y)

model.predict([[0,1,3400]]) # 3400 sqr ft home in west windsor

model.predict([[1,0,2800]]) # 2800 sqr ft home in robbinsville
```

Assignment No:6

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

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

dataset.head()

dataset.info()

X=dataset.iloc[:,:4].values
y=dataset["species"].values

y

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

from sklearn.preprocessing import StandardScaler,normalize
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_test)

from sklearn.naive_bayes import GaussianNB, MultinomialNB
model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
y_pred

model.predict([[-0.11774441,  2.19548765, -1.52234863, -1.34836924]])
#X_test[0]

from sklearn.metrics import confusion_matrix,precision_score,recall_score
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))

df = pd.DataFrame({'original Values':y_test, 'Predicted Values':y_pred})
df

print("Precision Score : ",precision_score(y_test, y_pred,pos_label='positive', average='micro'))
print("Recall Score : ",recall_score(y_test, y_pred, pos_label='positive',average='micro'))
```

Assignment No:7

```
import numpy as np
import pandas as pd
import sklearn

docs = pd.read_csv('example_train1.csv')
#text in column 1, classifier in column 2.
docs

# convert label to a numerical variable
docs['Class'] = docs.Class.map({'cinema':0, 'education':1})
docs

numpy_array = docs.to_numpy()
X = numpy_array[:,0]
Y = numpy_array[:,1]
Y = Y.astype('int')
print("X")
print(X)
print("Y")
print(Y

# create an object of CountVectorizer() class
```

```
from sklearn.feature_extraction.text import CountVectorizer
vec = CountVectorizer( )

# removing the stop words
vec = CountVectorizer(stop_words='english' )
vec.fit(X)
vec.vocabulary_

# printing feature names
print(vec.get_feature_names())
print(len(vec.get_feature_names()))

# another way of representing the features
X_transformed=vec.transform(X)
X_transformed

print(X_transformed)

# converting transformed matrix back to an array
# note the high number of zeros
X=X_transformed.toarray()
X

# converting matrix to dataframe
pd.DataFrame(X, columns=vec.get_feature_names())
```

Assignment No:8

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

dataset = sns.load_dataset('titanic')

dataset.head(20)

dataset.info()adult_maleadult_male

sns.distplot(dataset['fare'])

#@sns.distplot(dataset['fare'], kde=False)
#sns.histplot(dataset['fare'], kde=False)
sns.histplot(dataset['fare'], kde=True)

#sns.distplot(dataset['fare'], kde=False, bins=10)
sns.histplot(dataset['fare'], kde=False, bins=10, log_scale=(0,10))

sns.jointplot(x='age', y='fare', data=dataset)
```

```
#sns.jointplot(x='age', y='fare', data=dataset, kind='reg')
#sns.jointplot(x='age', y='fare', data=dataset, kind='hex')
sns.jointplot(x='age', y='fare', data=dataset, kind='scatter')

sns.pairplot(dataset[['age','fare']])

dataset.isna()

#dataset['deck'].isnull()
dataset[dataset['deck'].isnull()]

dataset[dataset['age'].isnull()]

dataset.dropna()

dataset['sex'].value_counts()

dataset.info()

dataset['deck'].isnull().value_counts()

dataset['adult_male'].value_counts()

sns.barplot(x='sex', y='age', data=dataset)

sns.countplot(x='sex', data=dataset)

sns.boxplot(x='sex', y='age', data=dataset)

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

sns.stripplot(x='sex', y='age', data=dataset)

sns.swarmplot(x='sex', y='age', data=dataset)

plt.hist(x='age')
```

Assignment -9

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

dataset = sns.load_dataset('titanic')

dataset.head(20)
```

```
dataset.info()adult_maleadult_male

sns.distplot(dataset['fare'])

#@sns.distplot(dataset['fare'], kde=False)
#sns.histplot(dataset['fare'], kde=False)
sns.histplot(dataset['fare'], kde=True)

#sns.distplot(dataset['fare'], kde=False, bins=10)
sns.histplot(dataset['fare'], kde=False, bins=10, log_scale=(0,10))

sns.jointplot(x='age', y='fare', data=dataset)

#sns.jointplot(x='age', y='fare', data=dataset, kind='reg')
#sns.jointplot(x='age', y='fare', data=dataset, kind='hex')
sns.jointplot(x='age', y='fare', data=dataset, kind='scatter')

sns.pairplot(dataset[['age','fare']])

dataset.isna()

#dataset['deck'].isnull()
dataset[dataset['deck'].isnull()]

dataset[dataset['age'].isnull()]

dataset.dropna()

dataset['sex'].value_counts()

dataset.info()

dataset['deck'].isnull().value_counts()

dataset['adult_male'].value_counts()

sns.barplot(x='sex', y='age', data=dataset)

sns.countplot(x='sex', data=dataset)

sns.boxplot(x='sex', y='age', data=dataset)

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

sns.stripplot(x='sex', y='age', data=dataset)

sns.swarmplot(x='sex', y='age', data=dataset)

plt.hist(x='age')
```

Assignment 10

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

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

dataset.head()

dataset.info()

X=dataset.iloc[:,:4].values
y=dataset["species"].values
y

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

from sklearn.preprocessing import StandardScaler,normalize
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_test)

from sklearn.naive_bayes import GaussianNB, MultinomialNB
model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
y_pred

model.predict([[-0.11774441,  2.19548765, -1.52234863, -1.34836924]])
#X_test[0]

from sklearn.metrics import confusion_matrix,precision_score,recall_score
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))

df = pd.DataFrame({'original Values':y_test, 'Predicted Values':y_pred})
df

print("Precision Score : ",precision_score(y_test, y_pred,pos_label='positive', average='micro'))
print("Recall Score : ",recall_score(y_test, y_pred, pos_label='positive',average='micro'))
```