\*\*\*FISH\*\*\*

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

import seaborn as sns

import matplotlib.pyplot as plt

import statsmodels.api as sm

import statsmodels.formula.api as smf

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

data = pd.read\_csv('../input/fish-market/Fish.csv')

df = data.copy()

df.sample(10)

sns.barplot(x=sp.index, y=sp['Species']);

plt.xlabel('Species')

plt.ylabel('Counts of Species')

plt.show()

g = sns.pairplot(df, kind='scatter', hue='Species');

*# Dependant (Target) Variable:*

y = df1['Weight']

*# Independant Variables:*

X = df1.iloc[:,2:7]

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, random\_state=1)

reg = LinearRegression()

reg.fit(X\_train,y\_train)

\*\*\*diabetics\*\*\*

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

import warnings

warnings.filterwarnings('ignore')

*#Loading the dataset*

diabetes\_data = pd.read\_csv('../input/diabetes.csv')

*#Print the first 5 rows of the dataframe.*

diabetes\_data.head()

diabetes\_data\_copy = diabetes\_data.copy(deep = True)

diabetes\_data\_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = diabetes\_data\_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)

diabetes\_data\_copy['Glucose'].fillna(diabetes\_data\_copy['Glucose'].mean(), inplace = True)

diabetes\_data\_copy['BloodPressure'].fillna(diabetes\_data\_copy['BloodPressure'].mean(), inplace = True)

diabetes\_data\_copy['SkinThickness'].fillna(diabetes\_data\_copy['SkinThickness'].median(), inplace = True)

diabetes\_data\_copy['Insulin'].fillna(diabetes\_data\_copy['Insulin'].median(), inplace = True)

diabetes\_data\_copy['BMI'].fillna(diabetes\_data\_copy['BMI'].median(), inplace = True)

p = diabetes\_data.hist(figsize = (20,20))

p=sns.pairplot(diabetes\_data\_copy, hue = 'Outcome')

plt.figure(figsize=(12,10)) *# on this line I just set the size of figure to 12 by 10.*

p=sns.heatmap(diabetes\_data.corr(), annot=True,cmap ='RdYlGn') *# seaborn has very simple solution for heatmap*

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

X = pd.DataFrame(sc\_X.fit\_transform(diabetes\_data\_copy.drop(["Outcome"],axis = 1),),

columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age'])

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=1/3,random\_state=42, stratify=y)

from sklearn.neighbors import KNeighborsClassifier

test\_scores = []

train\_scores = []

for i **in** range(1,15):

knn = KNeighborsClassifier(i)

knn.fit(X\_train,y\_train)

train\_scores.append(knn.score(X\_train,y\_train))

test\_scores.append(knn.score(X\_test,y\_test))

\*\*\*GLASS\*\*\*

import warnings

warnings.filterwarnings('ignore')

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn import tree

from sklearn.preprocessing import StandardScaler

df=pd.read\_csv('../input/glass/glass.csv')

sns.countplot(x="Type", data=df, palette='bright')

plt.show()

cor = df.corr()

plt.figure(figsize=(9,6))

sns.heatmap(data = cor, annot = True, cmap = 'PiYG')

plt.show()

columns=list(df.columns)

columns.remove('Type')

X=df.drop('Type', axis=1)

Y=df['Type']

X\_train, X\_test, Y\_train, Y\_test=train\_test\_split(X, Y, test\_size=0.20, random\_state=42)

dt\_gini=DecisionTreeClassifier(criterion='gini')

dt\_gini.fit(X\_train, Y\_train)

\*\*\* startups \*\*\*

Linear regression

\*\*\* computer \*\*\*

import numpy as np *# linear algebra*

import pandas as pd

train\_data=pd.read\_csv("/kaggle/input/mobile-price-range-prediction-is2020/train\_data.csv")

test\_data=pd.read\_csv("/kaggle/input/mobile-price-range-prediction-is2020/test\_data.csv")

x=train\_data.drop(columns=['price\_range','id'])

x

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, random\_state = 50)

print(x\_train.shape)

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression(C=100,multi\_class='multinomial', random\_state=0,max\_iter=1000)

lr = lr.fit(x\_train, y\_train)

pred=lr.predict(x\_test)

print(lr.score(x\_train, y\_train))

print(lr.score(x\_test, y\_test))

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(x\_train, y\_train)

from sklearn.naive\_bayes import GaussianNB

nb = GaussianNB()

nb = nb.fit(x\_train, y\_train)

In [16]:

print(nb.score(x\_train, y\_train))

print(nb.score(x\_test, y\_test))

from sklearn.svm import SVC

svm = SVC(C=1.0, kernel='linear', random\_state=0)

svm = svm.fit(x\_train, y\_train)

In [23]:

print(svm.score(x\_train, y\_train))

print(svm.score(x\_test, y\_test))

final\_pred1=svm.predict(test\_data)

final\_pred1

\*\*\*tours and travels\*\*\*

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.decomposition import PCA

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix, roc\_curve, auc, accuracy\_score

df\_travel=pd.read\_csv('/kaggle/input/travel-insurance/travel insurance.csv')

df\_travel.head()

df\_travel=df\_travel.drop(['Gender'], axis=1)

df\_travel.head()

fig = plt.figure(figsize = (10, 5))

plt.hist(df\_travel['Age'])

plt.xlabel("Age")

plt.ylabel("Number of people")

plt.title("Distribution of Age")

plt.show()

sns.pairplot(df\_travel, hue="Claim")

fig, ax = plt.subplots(figsize=(15, 10))

sns.heatmap(df\_travel.corr(), annot=True, ax=ax)

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

preds = model.predict(X\_test)

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train.values.ravel())

preds = model.predict(X\_test)

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import OneHotEncoder

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

import pathlib

from sklearn.metrics import confusion\_matrix ,accuracy\_score

data = pd.read\_csv(data\_apth)

data['Potability'] = data['Potability'].astype('category')

data.fillna(value=data.median(), inplace=True)

\*\*\* waterQuality \*\*\*

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

df = pd.read\_csv('../input/water-potability/water\_potability.csv')

fig,ax = plt.subplots(figsize = (10,7))

sns.heatmap(df.corr(),annot = True)

train = df.sample(frac = 0.7,random\_state = 25)

test = df.sample(frac = 0.3,random\_state = 26)

sns.pairplot(train[['ph','Sulfate','Trihalomethanes']])

for feat **in** train.columns:

train[feat] = train[feat].fillna(train[feat].mean())

logistic regression

random forest classifier

model = RandomForestClassifier()

model.fit(X\_train\_smote,y\_train\_smote)

y\_pred = model.predict(X\_test)

\*\*\*Placement\*\*\*

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import matplotlib

%matplotlib inline

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv(r"../input/factors-affecting-campus-placement/Placement\_Data\_Full\_Class.csv")

sns.countplot(data=df,x='gender')

sns.countplot(data=df,x='hsc\_b')

sns.countplot(data=df,x='hsc\_s')

sns.displot(data=df,x='ssc\_b',col='gender',hue='gender')

sns.displot(data=df,x='hsc\_b',col='gender',hue='gender')

sns.histplot(data=df,x='ssc\_b',hue='status',kde=True)

sns.histplot(data=df,x='hsc\_b',hue='status',kde=True)

sns.heatmap(df.drop(['sl\_no','salary'],axis=1).corr(),cmap='Oranges',annot=True)

X1 = df.drop(columns=['sl\_no','salary','status'])

y1 = df['status']

X1\_train,X1\_test,y1\_train,y1\_test = train\_test\_split(X1,y1,test\_size=0.25)

random\_forest **=** RandomForestClassifier(n\_estimators**=**350, min\_samples\_leaf**=**2, random\_state**=**42)

random\_forest**.**fit(X\_train, y\_train)

y\_pred **=** random\_forest**.**predict(X\_test)

accuracy\_score(y\_test,y\_pred)

from sklearn.linear\_model import LogisticRegression

clf=LogisticRegression()

clf.fit(X\_train,y\_train)

from sklearn import tree

model\_2 = tree.DecisionTreeClassifier(criterion='entropy')

model\_2.fit(X\_train, y\_train)

model\_2.score(X\_train,y\_train)