HW2 EDWARD PRIYATNA

PART A

1. ARD.shape
   9140 rows 22 collumns

2. types_NRA= len(pd.unique(ARD['National Remoteness Areas']))
   print('There are',types_NRA,'types of NRA')
   Ans: There are 6 types of NRA

   types_SA4= len(pd.unique(ARD['SA4 Name 2016']))
   print('There are',types_NRA,'SA4 Name 2016')
   Ans: There are 6 SA4 Name 2016

   types_LGA= len(pd.unique(ARD['National LGA Name 2017']))
   print('There are',types_NRA,'National LGA Name 2017')
   Ans: There are 6 National LGA Name 2017

3. ARD = ARD.replace('Unspecified', np.nan, regex=True) #replacing unspecified values with nan
   ARD
   ARD = ARD.replace('Undetermined', np.nan, regex=True)
   ARD
   ARD.isna().any(axis=1).sum()
   A3 1 Ans: 2302 rows with missing values

   ARD['YYYYMM']=ARD['YYYYMM'].map(str) #turning YYYYMM to string
   ARD['Month']=ARD['YYYYMM'].str[-2:]
   ARD
   null_ARD=ARD[ARD.isnull().any(axis=1)]
   null_ARD
   null_ARD['Month'].unique()
   A3 2 Ans: It looks like every month has a missing value

   ARD=ARD.dropna()
   ARD
   A3 3 answer

   ARD=ARD.drop_duplicates()
   ARD
   A3 4 answer

4. ARD['Month'].value_counts(ascending=True)
   A4 answer: month with most crashes are February and November

5. 1A
   ARD['Year']=ARD['YYYYMM'].str[:4]
   ARD
   ARD_crash_car_driver=ARD.loc[(ARD['Road User']=='Car driver')]
   ARD_crash_car_driver

ne	Crash Type	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	Road User	...	National Remoteness Areas	SA4 Name 2016	National LGA Name 2017	National Road Type	Christmas Period	Easter Period	Age Group	Time of day	Month	Year
00	Single	No	No	No	Car driver	...	Major Cities of Australia	Logan - Beaudesert	Logan (C)	Local Road	No	No	40_to_64	Night	09	2021
00	Single	No	No	No	Car driver	...	Inner Regional Australia	Adelaide - Central and Hills	Adelaide Hills (DC)	Sub-Arterial Road	No	No	17_to_25	Night	09	2021
00	Single	No	No	No	Car driver	...	Inner Regional Australia	Central Coast	Central Coast	Arterial Road	No	No	26_to_39	Night	09	2021
00	Multiple	No	No	No	Car driver	...	Major Cities of Australia	Ipswich	Ipswich (C)	National or State Highway	No	No	26_to_39	Night	09	2021
00	Single	No	No	No	Car driver	...	Outer Regional Australia	Wide Bay	South Burnett (R)	Sub-Arterial Road	No	No	40_to_64	Day	09	2021
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
00	Single	No	No	Yes	Car driver	...	Major Cities of Australia	Adelaide - Central and Hills	Unley (C)	National or State Highway	No	No	40_to_64	Night	01	2014
00	Multiple	No	No	Yes	Car driver	...	Outer Regional Australia	South Australia - South East	Wattle Range (DC)	National or State Highway	No	No	26_to_39	Day	01	2014
00	Multiple	No	No	No	Car driver	...	Inner Regional Australia	South Australia - South East	Alexandrina (DC)	National or State Highway	No	No	26_to_39	Day	01	2014
00	Multiple	No	Yes	No	Car driver	...	Outer Regional Australia	South Australia - South East	The Coorong (DC)	National or State Highway	No	No	40_to_64	Day	01	2014
00	Single	No	No	No	Car	...	Remote	Western Australia -	Esperance	National or State	No	No	75_or_older	Night	01	2014

   1B
   ARD_crash_car_driver_year_month=ARD_crash_car_driver.groupby(['Year', 'Month']).size()
   ARD_crash_car_driver_year_month
   ARD_crash_car_driver_year_month.to_csv('ARD_crash_car_driver_year_month.csv')
   ARD_crash_car_driver_year_month=pd.read_csv('ARD_crash_car_driver_year_month.csv')
   ARD_crash_car_driver_year_month

   Ans: To see it fully, open 'ARD_crash_car_driver_year_month.csv'. Year is col 0, month is col 1, crash for each month by year is col 2.

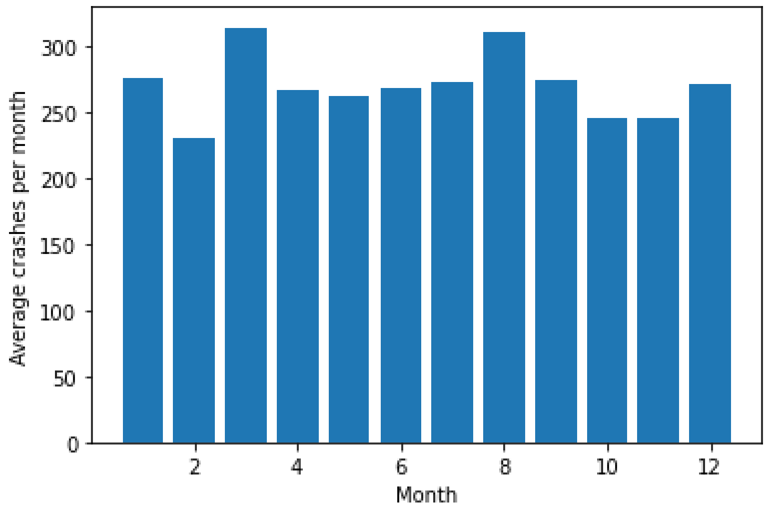
	Year	Month	0
0	2014	1	7
1	2014	2	10
2	2014	3	12
3	2014	4	9
4	2014	5	6
5	2014	6	14
6	2014	7	11
7	2014	8	15
8	2014	9	10
9	2014	10	12
10	2014	11	12
11	2014	12	18
12	2015	1	41
13	2015	2	27
14	2015	3	27
15	2015	4	26
16	2015	5	43
17	2015	6	39
18	2015	7	33
19	2015	8	39
20	2015	9	38
21	2015	10	42

1C

```
ARD_crash_car_driver_month_count=ARD_crash_car_driver['Month'].value_counts()
.sort_index()
ARD_crash_car_driver_month_count
ARD_crash_car_driver_month_AVG=ARD_crash_car_driver_month_count/7
ARD_crash_car_driver_month_AVG
```

```
01    39.285714
02    32.857143
03    44.857143
04    38.142857
05    37.428571
06    38.285714
07    38.857143
08    44.285714
09    39.142857
10    35.142857
11    35.000000
12    38.714286
Name: Month, dtype: float64
```

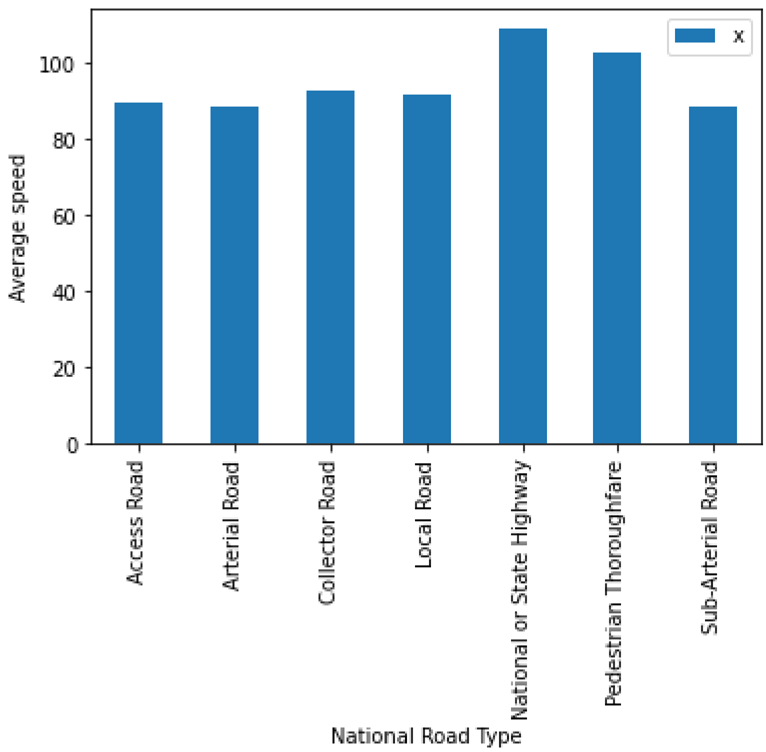
2



3. There is not much variance between crashes per month.

A6

1. Avg_speed_CD=ARD_crash_car_driver.groupby(['National Road Type'])['Speed'].agg(x='mean')
   Avg_speed_CD
   Avg_speed_CD.plot.bar(rot=0)
   plt.xticks(rotation=90)
   plt.ylabel('Average speed')



2. ARD['Age'].unique()
   ARD=ARD.copy()
   ARD.loc[ARD['Age']<ARD['Driving experience'],'Age']=0 #replacing counter intuitive values with 0
   ARD

```
ARD=ARD.copy()
ARD.loc[ ARD['Age']== -999,'Age'  ]=0 #replacing age=-999 to 0
ARD
```

A7

1. ARD7A={'Road User':ARD['Road User'],'Age':ARD['Age'],'Speed':ARD['Speed'],'Driving experience':ARD['Driving experience']}

```
ARD7A=pd.DataFrame(ARD7A)
ARD7A
def group_vehicle_driver(vehicle_driver):
   ARD7A_vehicle_driver=ARD7A.loc[(ARD7A['Road User']==vehicle_driver)]
   return ARD7A_vehicle_driver
road_user=ARD7A['Road User'].unique()
road_user

ARD7A_MR=group_vehicle_driver('Motorcycle rider')
ARD7A_MR
print('Motorcycle rider correlation')
print(ARD7A_MR.corr())
```

```
Motorcycle rider correlation
                         Age     Speed  Driving experience
Age                 1.000000  0.003875            0.906471
Speed               0.003875  1.000000           -0.002233
Driving experience  0.906471 -0.002233            1.000000
```

```
ARD7A_CD=group_vehicle_driver('Car driver')
ARD7A_CD
print('Car driver correlation')
print(ARD7A_CD.corr())
```

```
Car driver correlation
                         Age     Speed  Driving experience
Age                 1.000000  0.014208            0.938079
Speed               0.014208  1.000000            0.012633
Driving experience  0.938079  0.012633            1.000000
```

```
ARD7A_PC=group_vehicle_driver('Pedal cyclist')
ARD7A_PC
print('Pedal cyclist correlation')
print(ARD7A_PC.corr())
```

```
Pedal cyclist correlation
                         Age     Speed  Driving experience
Age                 1.000000 -0.058045            0.785747
Speed              -0.058045  1.000000           -0.113703
Driving experience  0.785747 -0.113703            1.000000
```

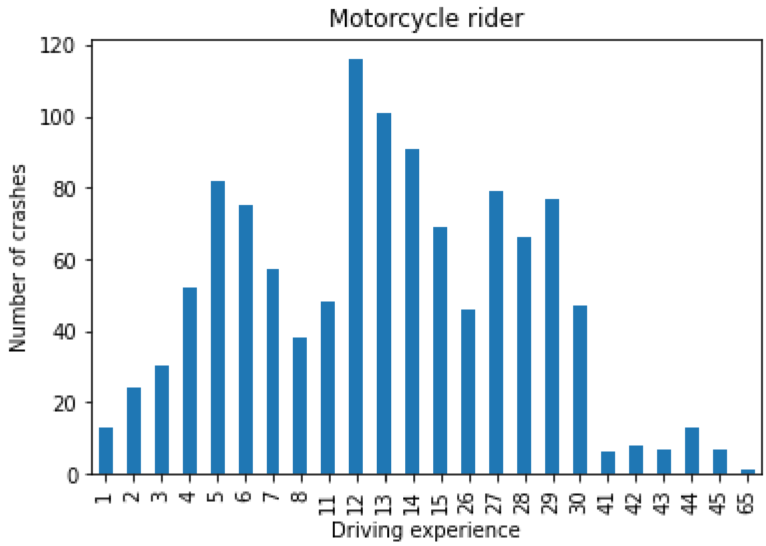
```
ARD7A_OVD=group_vehicle_driver('Other vehicle driver')
ARD7A_OVD
print('Other vehicle driver correlation')
print(ARD7A_OVD.corr())
```

```
Other vehicle driver correlation
                         Age     Speed  Driving experience
Age                 1.000000 -0.122975            0.933673
Speed              -0.122975  1.000000           -0.041656
Driving experience  0.933673 -0.041656            1.000000
```

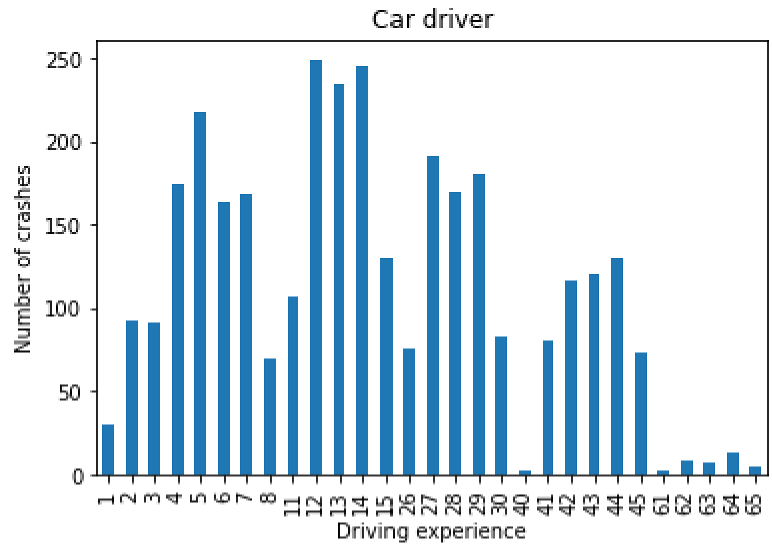
Age and driving experience has the most correlation

2.

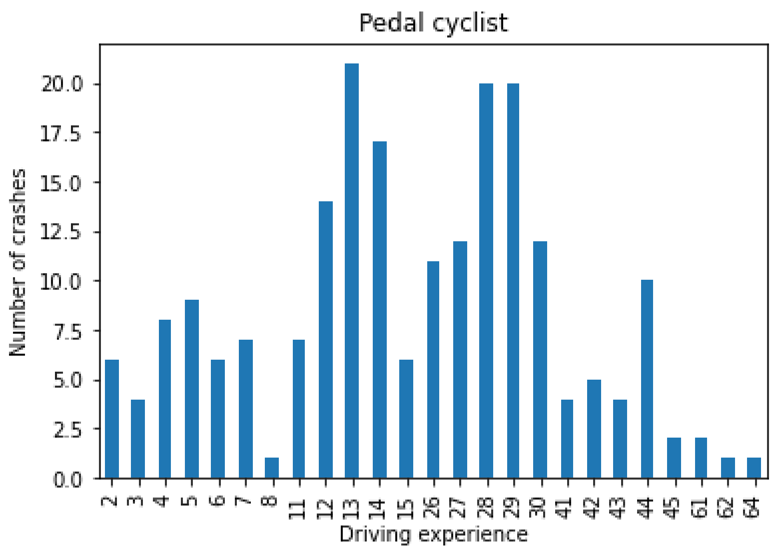
```
ARD7B_MR=ARD7A_MR['Driving experience'].value_counts().sort_index()
ARD7B_MR
ARD7B_MR.plot(kind='bar')
plt.xlabel("Driving experience")
plt.ylabel("Number of crashes")
plt.title("Motorcycle rider")
plt.show()
```



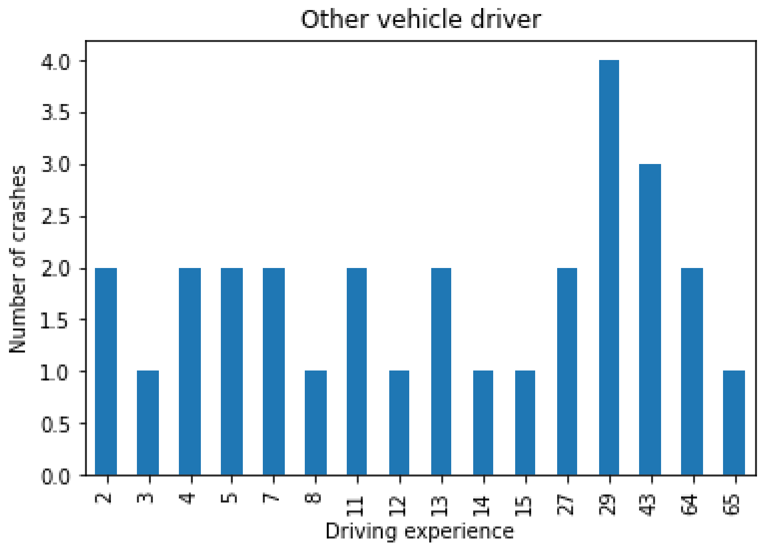

```
ARD7B_CD=ARD7A_CD['Driving experience'].value_counts().sort_index()
ARD7B_CD
ARD7B_CD.plot(kind='bar')
plt.xlabel("Driving experience")
plt.ylabel("Number of crashes")
plt.title("Car driver")
plt.show()
```




```
ARD7B_PC=ARD7A_PC['Driving experience'].value_counts().sort_index()
ARD7B_PC
ARD7B_PC.plot(kind='bar')
plt.xlabel("Driving experience")
plt.ylabel("Number of crashes")
plt.title("Pedal cyclist")
plt.show()
```



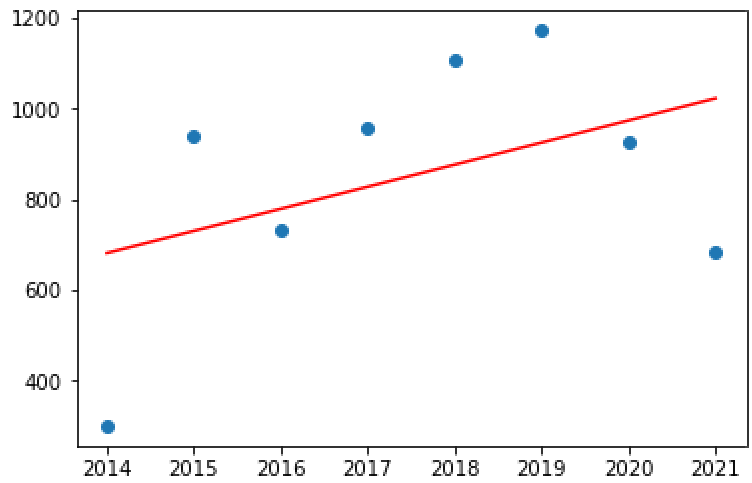

```
ARD7B_OVD=ARD7A_OVD['Driving experience'].value_counts().sort_index()
ARD7B_OVD
ARD7B_OVD.plot(kind='bar')
plt.xlabel("Driving experience")
plt.ylabel("Number of crashes")
plt.title("Other vehicle driver")
plt.show()
```



A8

```
1. crash_per_year=ARD['Year'].value_counts().sort_index()

crash_per_year

crash_per_year = pd.DataFrame(ARD['Year'].value_counts().reset_index().values, columns=["Year", "crashes"])

crash_per_year

crash_per_year=crash_per_year.sort_values('Year')

crash_per_year

%matplotlib inline

plt.xlabel('Year')

plt.ylabel('crashes')

#plt.scatter(df.area,df.price,color='red',marker='+')

plt.scatter(crash_per_year.Year,crash_per_year.crashes,color='red',marker='+')

crash_per_year['Year'] = crash_per_year['Year'].astype(int)

X81 = crash_per_year.iloc[:, 0].values.reshape(-1, 1)

Y81 = crash_per_year.iloc[:, 1].values.reshape(-1, 1)

linear_regressor = LinearRegression()

linear_regressor.fit(X81, Y81)

Y_pred_81 = linear_regressor.predict(X81)

plt.scatter(X81, Y81)

plt.plot(X81, Y_pred_81, color='red')
```

plt.show()



2. Year = crash_per_year.drop('crashes',axis='columns')

Year

crashes=crash_per_year['crashes']

crashes

reg = linear_model.LinearRegression()

reg.fit(Year,crashes)

reg.predict([[2022]])

Answer: There will be 1072.07142857 crashes in 2022.

3. x_np= crash_per_year.iloc[:, 0].values

y_np = crash_per_year.iloc[:, 1].values

x_np=x_np.reshape(-1, 1)

x_np

y_np

lin = LinearRegression()

lin.fit(x_np, y_np)

poly = PolynomialFeatures(degree = 2)

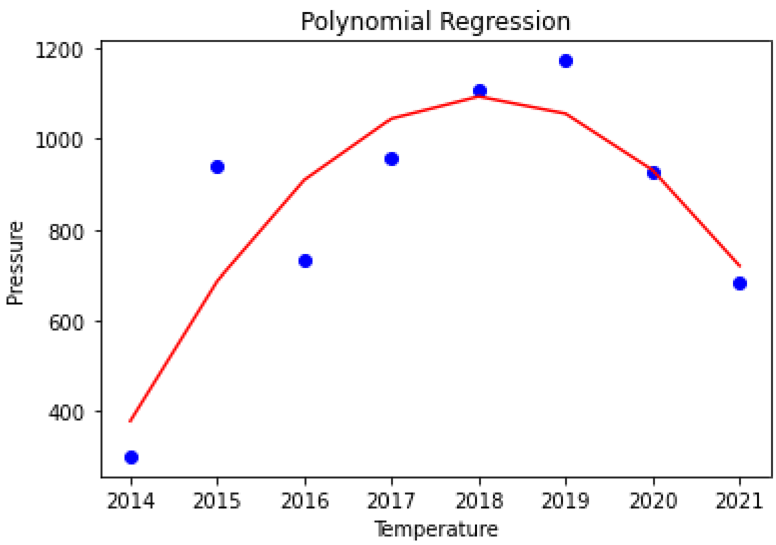
x_poly = poly.fit_transform(x_np)

poly.fit(x_poly, y_np)

lin2 = LinearRegression()

lin2.fit(x_poly, y_np)

```
plt.scatter(x_np, y_np, color = 'blue')

plt.plot(x_np, lin.predict(x_np), color = 'red')
plt.title('Linear Regression')
plt.xlabel('Year')
plt.ylabel('Crashes')
plt.show()

plt.scatter(x_np, y_np, color = 'blue')
plt.plot(x_np, lin2.predict(poly.fit_transform(x_np)), color = 'red')
plt.title('Polynomial Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')
plt.show() #this is a 2 degree polynomial
```



Answer: Better model is 2nd degree polynomial. Look at the line, it fits the dots better.

4. pred2 = 2022 #predicted value for 2022 using polynomial is 324.57 crashes

pred2array = np.array([[pred2]])

lin2.predict(poly.fit_transform(pred2array))

Polynomial Regression predicted there will be 422.42857134 crashes in 2022.

A8

```
DE_and_Age={'Driving experience':ARD['Driving experience'],'Age':ARD['Age']}

DE_and_Age=pd.DataFrame(DE_and_Age)

DE_and_Age

DE_and_Age_weird=DE_and_Age.loc[ (DE_and_Age['Driving experience']>
DE_and_Age['Age'] )]

DE_and_Age_weird

DE_and_Age = DE_and_Age.drop((DE_and_Age_weird.index))

DE_and_Age

plt.xlabel('Driving experience')

plt.ylabel('Age')

plt.scatter(DE_and_Age['Driving
experience'],DE_and_Age['Age'],color='red',marker='+')

new_DEA = DE_and_Age.drop('Age',axis='columns')

new_DEA

Age = DE_and_Age.Age

Age

reg = linear_model.LinearRegression()

reg.fit(new_DEA,Age)

DE_and_Age_weird=DE_and_Age_weird.drop('Age',axis='columns')

DE_and_Age_weird

age_predict=reg.predict(DE_and_Age_weird)

age_predict

DE_and_Age_weird['Age']=age_predict

DE_and_Age_weird.head(30)
```

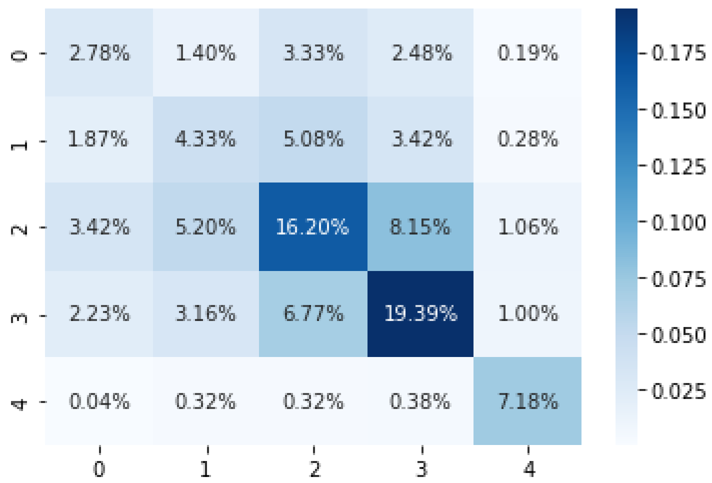
This is some sample of predicted age based on driving experience

	Driving experience	Age
8	3	22.349745
40	40	75.619456
46	62	107.293338
90	40	75.619456
107	63	108.733060
283	62	107.293338
371	4	23.789467
608	1	19.470301
713	65	111.612504
825	4	23.789467
1227	2	20.910023
1268	64	110.172782
1299	3	22.349745
1314	4	23.789467
1315	3	22.349745
1443	3	22.349745
1696	4	23.789467
1717	4	23.789467
1760	2	20.910023
2222	3	22.349745
2494	2	20.910023
2723	5	25.229189

B1

```
from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

from sklearn import metrics

song_data=pd.read_csv('song_data.csv')

song_data

sd=song_data.drop(['song_name'],axis=1 )

sd

X = sd.drop('song_popularity', axis=1)

y = song_data['song_popularity']

X

Y
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

classifier = DecisionTreeClassifier()

classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

y_pred

from sklearn.metrics import classification_report, confusion_matrix

cf_matrix = confusion_matrix(y_test, y_pred)

print(cf_matrix)

print(classification_report(y_test, y_pred))

import seaborn as sns

sns.heatmap(cf_matrix, annot=True)

sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Blues')
```



X axis actual values, Y axis predicted values. For each index we add plus one. For example index 0 is actually 1. We can see that this model is very bad at predicting song popularity. Since a lot of the values are predicted wrongly.

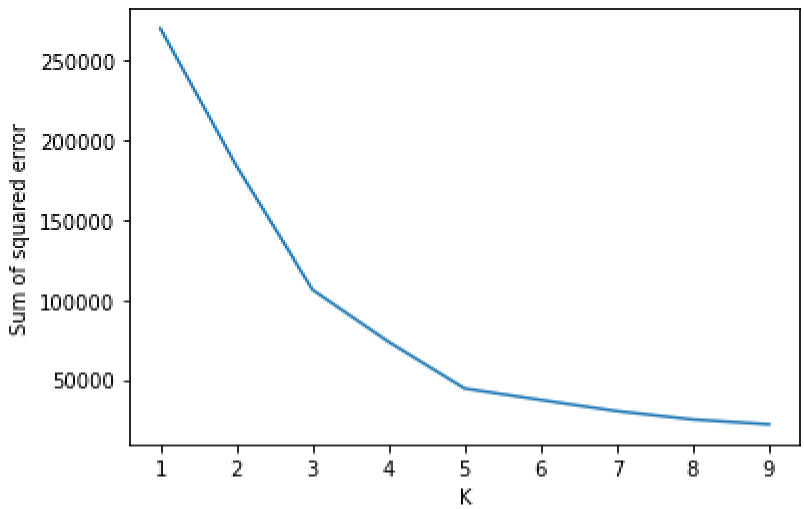
B2

```
from sklearn.cluster import KMeans

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

%matplotlib inline

mc = pd.read_csv("Mall_Customers.csv")
```

```
mc.head()
```

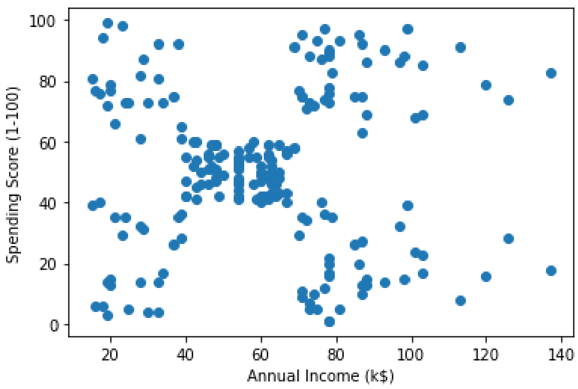
PLOTTING BASED ON ANNUAL INCOME AND SPENDING SCORE, cluster=5

```
plt.scatter(mc['Annual Income (k$)'],mc['Spending Score (1-100)'])

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

sse = [] #sum of squared error

k_rng = range(1,10)

for k in k_rng:

    km = KMeans(n_clusters=k)

    km.fit(mc[['Annual Income (k$)','Spending Score (1-100)']])

    sse.append(km.inertia_)

plt.xlabel('K') #plotting elbow plot

plt.ylabel('Sum of squared error')

plt.plot(k_rng,sse) #there are 5 clusters
```

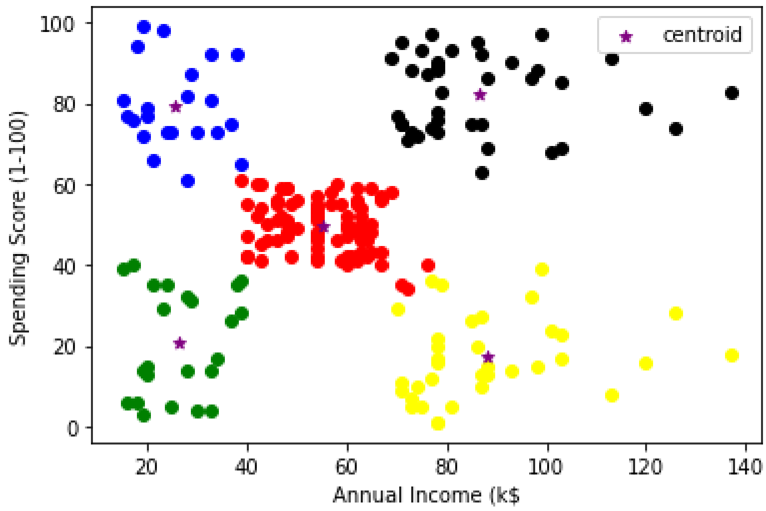


Plotting elbow plot. There are 5 clusters.

```
plt.scatter(mc['Annual Income (k$)'],mc['Spending Score (1-100)'])

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')
```

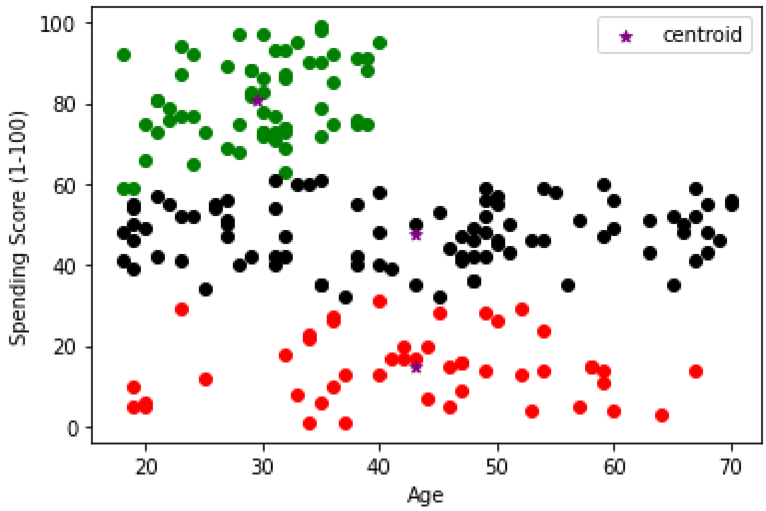



```
km = KMeans(n_clusters=5)
y_predicted = km.fit_predict(mc[['Annual Income (k$)','Spending Score (1-100)']])
y_predicted
mc['cluster']=y_predicted
mc.head(30)
km.cluster_centers_
mc1=mc[mc.cluster==0]
mc2=mc[mc.cluster==1]
mc3=mc[mc.cluster==2]
mc4=mc[mc.cluster==3]
mc5=mc[mc.cluster==4]
plt.scatter(mc1['Annual Income (k$)'],mc1['Spending Score (1-100)'],color='green')
plt.scatter(mc2['Annual Income (k$)'],mc2['Spending Score (1-100)'],color='red')
plt.scatter(mc3['Annual Income (k$)'],mc3['Spending Score (1-100)'],color='black')
plt.scatter(mc4['Annual Income (k$)'],mc4['Spending Score (1-100)'],color='blue' )
plt.scatter(mc5['Annual Income (k$)'],mc5['Spending Score (1-100)'],color='yellow' )
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='
*',label='centroid')
plt.xlabel('Annual Income (k$')
plt.ylabel('Spending Score (1-100)')
plt.legend()
```



PLOTTING BASED ON AGE AND SPENDING SCORE, cluster=3

```
km = KMeans(n_clusters=3)

y_predicted = km.fit_predict(mc[['Age','Spending Score (1-100)']])

y_predicted

mc['cluster']=y_predicted

mc.head()

km.cluster_centers_

mc1=mc[mc.cluster==0]

mc2=mc[mc.cluster==1]

mc3=mc[mc.cluster==2]

plt.scatter(mc1['Age'],mc1['Spending Score (1-100)'],color='green')

plt.scatter(mc2['Age'],mc2['Spending Score (1-100)'],color='red')

plt.scatter(mc3['Age'],mc3['Spending Score (1-100)'],color='black')

plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='centroid')

plt.xlabel('Age')

plt.ylabel('Spending Score (1-100)')

plt.legend()
```



After plotting 2 types of data. It seems that plotting based on age and spending score is more useful. Based on the plot, we can see that younger people spend more than older.

dataset link: https://www.kaggle.com/datasets/nelakurthisudheer/mall-customer-segmentation?resource=download