Table of Contents

Chapter 1- (Regression)

1) Introduction:

- Data Taken: Adult Censous Income(Regression)

This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)).

- Breakdown of the Problem Statement:

  The prediction task is to determine whether a person makes over $50K a year.

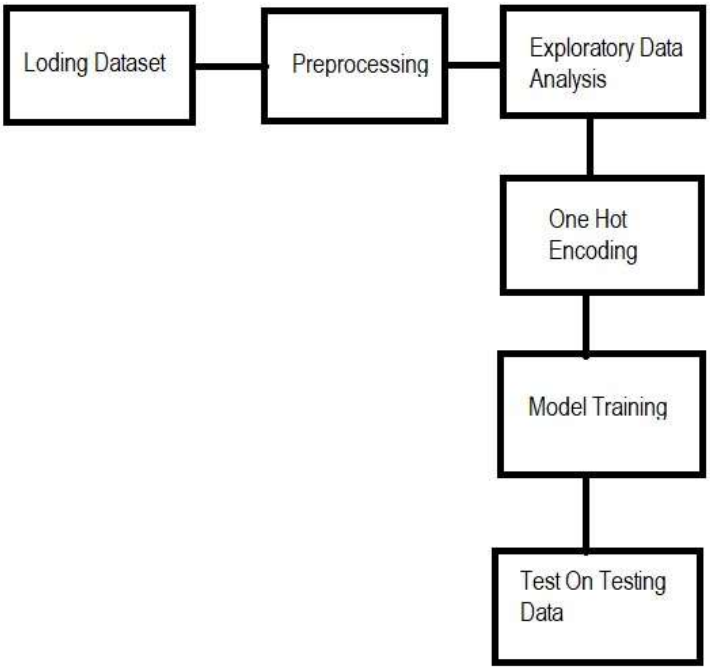
- Regression:

  - Regression is a technique for investigating the relationship between independent variables or features and a dependent variable or outcome. It's used as a method for predictive modelling in machine learning, in which an algorithm is used to predict continuous outcomes.

  - Solving regression problems is one of the most common applications for machine learning models, especially in supervised machine learning. Algorithms are trained to understand the relationship between independent variables and an outcome or dependent variable. The model can then be leveraged to predict the outcome of new and unseen input data, or to fill a gap in missing data.

  - Regression analysis is an integral part of any forecasting or predictive model, so is a common method found in machine learning powered predictive analytics. Alongside classification, regression is a common use for supervised machine learning models. This approach to training models required labelled input and output training data. Machine learning regression models need to understand the relationship between features and outcome variables, so accurately labelled training data is vital.

2) Purpose Methodology:



2.1) Loading Dataset
2.2) Preprocessing
2.3) Exploratory Data Analysis
2.4) Model Training
2.5) Test On Testing Data

2.1) Loading the Datasets

Python Command is used to Load the data.
->Import pandas as pd

->Dataset name=”train.csv”

->head(): Used to show First Five Rows

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df = pd.read_csv('adultcensousincome.csv')
df.head(10)
```

Out[3]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.co
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	United-
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	United-
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0	3770	45	United-

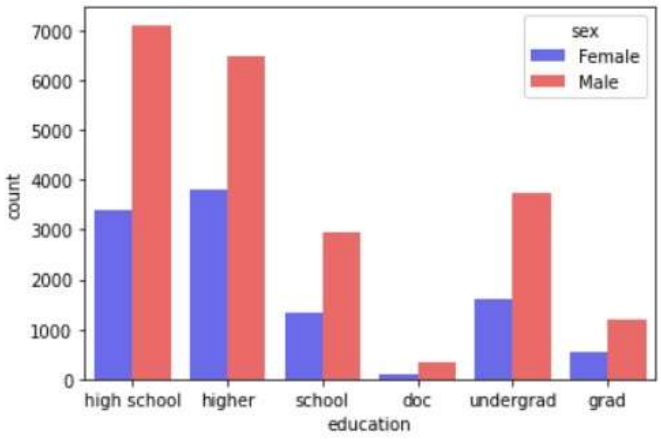
○ 2.2) Preprocessing

In [16]:
```
#Replace the value
df['workclass'] = df['workclass'].replace('?', 'Private')
df['occupation'] = df['occupation'].replace('?', 'Prof-specialty')
df['native.country'] = df['native.country'].replace('?', 'United-States')
df.head(10)
```

Out[16]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.co
0	90	Private	77053	HS-grad	9	Widowed	Prof-specialty	Not-in-family	White	Female	0	4356	40	United-
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-
2	66	Private	186061	Some-college	10	Widowed	Prof-specialty	Unmarried	Black	Female	0	4356	40	United-
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0	3770	45	United-
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	0	3770	40	United-
7	74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative	White	Female	0	3683	20	United-

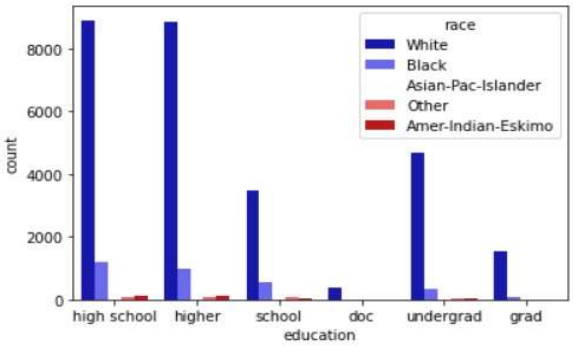
```
In [26]: 1 sns.countplot(df['education'], hue='sex', data=df, palette='seismic');
```



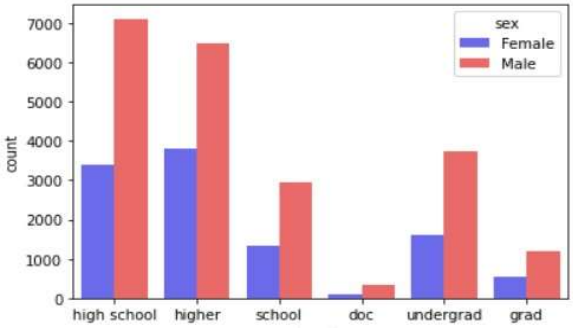
2.3) Exploratory Data Analysis

We can see that high school males are maximum and higher female are maximum.

```
In [25]: 1 sns.countplot(df['education'], hue='race', data=df, palette='seismic');
```




```
In [26]: 1 sns.countplot(df['education'], hue='sex', data=df, palette='seismic');
```



2.4) Model Training

Logistic Regression

```
]:
1  from sklearn.linear_model import LogisticRegression
2  from sklearn.metrics import accuracy_score
3
4  lr = LogisticRegression()
5
6  model = lr.fit(X_train, y_train)
7  prediction = model.predict(X_test)
8
9  print("Acc on training data: {:,.3f}".format(lr.score(X_train, y_train)))
10 print("Acc on test data: {:,.3f}".format(lr.score(X_test, y_test)))
```

```
Acc on training data: 0.839
Acc on test data: 0.836
```

Accuracy: - 83.6

Lasso Regression

Lasso

```
]:
1  import pandas as pd
2  import numpy as np
3  from sklearn.linear_model import Lasso
4  lr=Lasso()
5  lr.fit(X_train,y_train)
6  y_pred=lr.predict(X_test)
7  from sklearn.metrics import accuracy_score,r2_score
8  mse=np.square(np.subtract(y_test,y_pred)).mean()
9  r2=r2_score(y_test,y_pred)
10 print('MSE=',mse)
11 print('R2-Score',r2)
12 print(lr.coef_)
13 print(lr.intercept_)
```

```
MSE= 0.18338036279642306
R2-Score -1.2934752724236276e-05
[ 0.  0. -0.  0.  0. -0.  0. -0.  0.  0.  0.  0.  0.  0.]
0.24034749034749034
```

Accuracy: - 24.03

Ridge Model :

Ridge

[32]:
```
from sklearn.linear_model import Ridge
lr=Ridge()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
from sklearn.metrics import accuracy_score,r2_score
mse=np.square(np.subtract(y_test,y_pred)).mean()
r2=r2_score(y_test,y_pred)
print('MSE=',mse)
print('R2-Score',r2)
print(lr.coef_)
print(lr.intercept_)
```

```
MSE= 0.11813817454957452
R2-Score 0.35576688345414265
[ 0.06941406 -0.01369585  0.00754383  0.00626905  0.10078593 -0.12177695
  0.00276334  0.0132017   0.01214579  0.01582371  0.10500508  0.04965196
  0.04774843 -0.00240331]
0.24034293994797912
```

Accuracy: 35.57

- ElasticNet Model:

ElasticNet

:
```
from sklearn.linear_model import ElasticNet
lr=ElasticNet()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
from sklearn.metrics import accuracy_score,r2_score
mse=np.square(np.subtract(y_test,y_pred)).mean()
r2=r2_score(y_test,y_pred)
print('MSE=',mse)
print('R2-Score',r2)
print(lr.coef_)
print(lr.intercept_)
```

```
MSE= 0.18338036279642306
R2-Score -1.2934752724236276e-05
[ 0.  0. -0.  0.  0. -0.  0. -0.  0.  0.  0.  0.  0.  0.]
0.24034749034749034
```

- Accuracy: -1.29

In [68]:
```
# Predict the sale price On Ridge model
y_predicted_r=r.predict(x_test)
y_predicted_r
```

Out[68]:
```
array([2850.68068344, 1599.18461816, 3427.05687563, ..., 1228.83187078,
       4228.77031002, 1731.62441873])
```

Decision Tree Regressor

In [37]:
```
from sklearn.tree import DecisionTreeRegressor
lr=DecisionTreeRegressor()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
from sklearn.metrics import accuracy_score,r2_score
mse=np.square(np.subtract(y_test,y_pred)).mean()
r2=r2_score(y_test,y_pred)
print('MSE=',mse)
print('R2-Score',r2)
```

```
MSE= 0.18783908281297984
R2-Score -0.02432730310168152
```

In [38]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
print(confusion_matrix(y_test, y_pred))
```

```
[[6482  924]
 [ 911 1452]]
```

In [39]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      7406
           1       0.61      0.61      0.61      2363

    accuracy                           0.81      9769
   macro avg       0.74      0.74      0.74      9769
weighted avg       0.81      0.81      0.81      9769
```

So, Highest Accuracy is of Decision Tree Regressor.

Result:-

Model Accuracy:
Logistic Regression: 84.6
Lasso: 24.3
Ridge: 35.57
Elastic Net: -1.29
Decision Tree: 88

Decision Tree Regressor has Highest Accuracy.

3) Conclusion:

- We train our model on Decision Tree Regressor.
- Now we predict value by fit the testing data in it.

4) Reference(if any)

https://www.kaggle.com/datasets/uciml/adult-censusincome

Chapter 2-(Classification)

1) Introduction:

- Data Taken: Titanic Dataset(Classification)

The sinking of the Titanic is one of the most infamous shipwrecks in history.

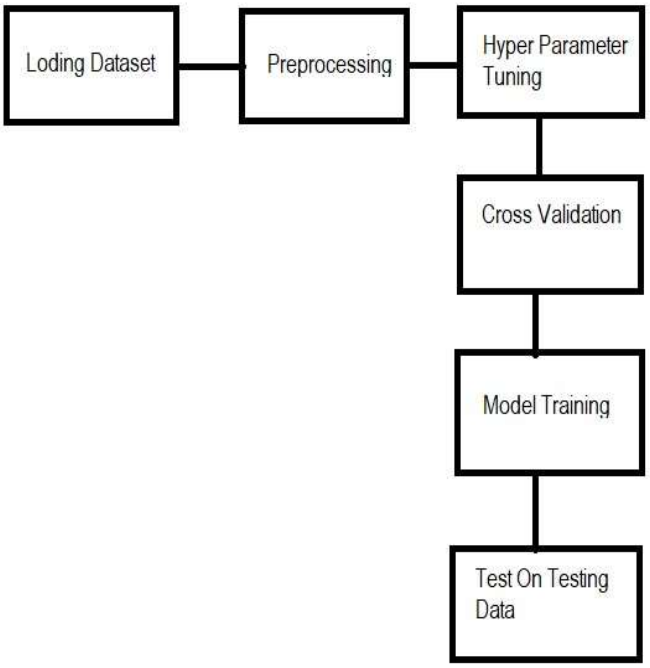
On April 15, 1912, during her maiden voyage, the widely considered "unsinkable" RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren't enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

- Classification:
  - In machine learning, classification is a supervised learning concept which basically categorizes a set of data into classes. The most common classification problems are – speech recognition, face detection, handwriting recognition, document classification, etc.
  - It can be either a binary classification problem or a multi-class problem too. There are a bunch of machine learning algorithms for

classification in machine learning. Let us take a look at those classification algorithms in machine learning.

Purpose Methodology:



- Loading Dataset
- Preprocessing
- Hyper parameter Tuning
- Cross validation
- Model training

- Test on testing data

2.1) Loading the Datasets

Python Command is used to Load the data.
->Import pandas as pd

->Dataset name=”train.csv”

->head(): Used to show First Five Rows

Loading Dataset

```
In [1]: 1 import pandas as pd
In [2]: 1 df=pd.read_csv("train.csv")
In [3]: 1 df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

2.2) Preprocessing

- ✦ Droping NULL Values From the Dataset

Preprocessing

```
In [4]: 1 df.drop(["Name","Ticket","Fare"],axis="columns",inplace=True)

In [5]: 1 df.isnull().sum()

Out[5]: PassengerId      0
        Survived         0
        Pclass           0
        Sex              0
        Age            177
        SibSp            0
        Parch            0
        Cabin          687
        Embarked         2
        dtype: int64

In [6]: 1 df.drop("Cabin",axis="columns",inplace=True)

In [7]: 1 df.isnull().sum()

Out[7]: PassengerId      0
        Survived         0
        Pclass           0
        Sex              0
        Age            177
        SibSp            0
        Parch            0
        Embarked         2
        dtype: int64
```

O 2.3) Hyper Parameter Tuning

- It is used to select the best perameters to train the model.
- We use GridSearchCV

In [36]:
```
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
```

In [38]:
```
model_params={
    'LogisticRegression':{
        'model':LogisticRegression(),
        'params':{
            'C':[1,10,20],
            'penalty':['l1','l2']
        }
    },
    'DecisionTreeClassifier':{
        'model':DecisionTreeClassifier(),
        'params':{
            'criterion':['gini', 'entropy']
        }
    },
    'random_forest_classifier':{
        'model':RandomForestClassifier(),
        'params':{
            'n_estimators':[1,10,20]
        }
    },
    'SVC':{
        'model':SVC(),
        'params':{
            'kernel':['linear', 'rbf'],
            'C':[1,5,10]
        }
    }
}
```

2.4) Model Training

- As SCV gives best result so train the model on SVC.
- And save that model in pickle.

```
In [40]:
scores_df=pd.DataFrame(scores,columns=["model","best_score","best_params"])
scores_df
```

Out[40]:

	model	best_score	best_params
0	LogisticRegression	0.792367	{'C': 1, 'penalty': 'l2'}
1	DecisionTreeClassifier	0.793541	{'criterion': 'entropy'}
2	random_forest_classifier	0.797991	{'n_estimators': 10}
3	SVC	0.810320	{'C': 10, 'kernel': 'rbf'}

Model Training

So we see SVC have batter results

```
In [52]:
model=SVC(C=10,kernel='rbf')
model.fit(x,y)
```

```
Out[52]: SVC(C=10)
```

O 2.5) Test Model on Testing data •

Test Model on Testing data.

- Save this result in .csv format.

```
In [53]:
from sklearn.model_selection import train_test_split
```

```
In [54]:
x_train,x_test,y_train,y_test=train_test_split(x,y)
```

```
In [55]:
x_train.shape
```

```
Out[55]: (668, 8)
```

```
In [56]:
x_test.shape
```

```
Out[56]: (223, 8)
```

```
In [57]:
y_predict=model.predict(x_test)
```

```
In [68]:
model.score(x_test,y_test)
```

```
Out[68]: 0.8251121076233184
```

3) Result:-

1) We perform classification on Titanic Dataset.
2) Model Accuracy:

- Logistic Regression: 79.23
- DecisionTreeClassifier: 79.35

- Random_forest_classifier: 79.79
- SVC: 94.65

3) SVC has Highest Accuracy.

4) Conclusion:

- We train our model on SVC.
- Now we predict value by fit the testing data in it.

5) Reference(if any)

https://www.kaggle.com/c/titanic

Chapter 3 - Clustering

1) Introduction:

- Data Taken: Turkiye Student Evaluation Data Set (Clustering)
- Attributes:

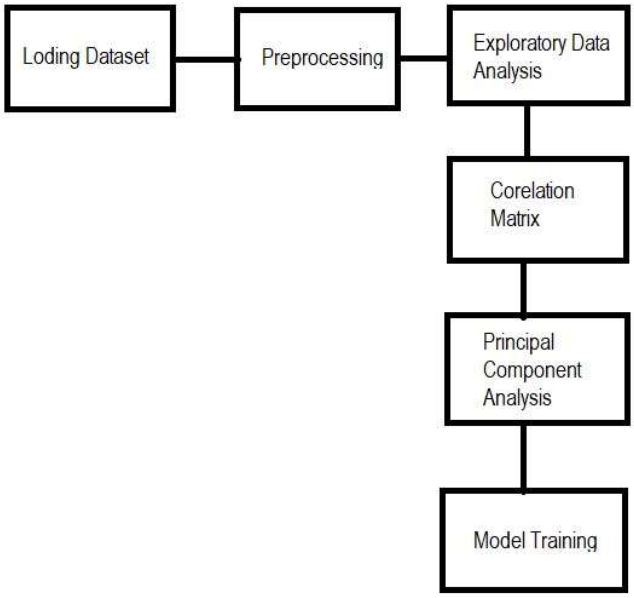
instr: Instructor's identifier; values taken from {1,2,3} class: Course code (descriptor); values taken from {1-13} repeat: Number of times the student is taking this course; values taken from {0,1,2,3,...} attendance: Code of the level of attendance; values from {0, 1, 2, 3, 4} difficulty: Level of difficulty of the course as perceived by the student; values taken from {1,2,3,4,5}
Q1: The semester course content, teaching method and evaluation system were provided at the start.
Q2: The course aims and objectives were clearly stated at the beginning of the period.
Q3: The course was worth the amount of credit assigned to it.
Q4: The course was taught according to the syllabus announced on the first day of class. Q5: The class discussions, homework assignments, applications and studies were satisfactory. Q6: The textbook and other courses resources were sufficient and up to date. Q7: The course allowed field work, applications, laboratory, discussion and other studies. Q8: The quizzes, assignments, projects

and exams contributed to helping the learning. Q9: I greatly enjoyed the class and was eager to actively participate during the lectures.

.
.
.
.

- Clustering:
  - Clustering or cluster analysis is a machine learning technique, which groups the unlabeled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points.
  - The objects with the possible similarities remain in a group that has less or no similarities with another group.

2) Purpose Methodology:



- Loading Dataset
- Preprocessing
- Exploratory Data Analysis
- Co-relation Matrix

- Principal Component Analysis
- Model Building

O 2.1) Loading data set

Python Command is used to Load the data.
Import pandas as pd
Dataset name=" turkiye-student-evaluation_generic.csv" head():
Used to show First Five Rows

Loading the dataset

In [1]:
```
import pandas as pd
df=pd.read_csv("turkiye-student-evaluation_generic.csv")
df.head()
```

Out[1]:

	instr	class	nb.repeat	attendance	difficulty	Q1	Q2	Q3	Q4	Q5	...	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27	Q28
0	1	2	1	0	4	3	3	3	3	3	...	3	3	3	3	3	3	3	3	3	3
1	1	2	1	1	3	3	3	3	3	3	...	3	3	3	3	3	3	3	3	3	3
2	1	2	1	2	4	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
3	1	2	1	1	3	3	3	3	3	3	...	3	3	3	3	3	3	3	3	3	3
4	1	2	1	0	1	1	1	1	1	1	...	1	1	1	1	1	1	1	1	1	1

5 rows × 33 columns

O 2.2) Preprocessing

- There is no NULL Value in the dataset.
- And all the values are in integer type. • So there is no need for preprocessing.

In [6]:
```
df.describe()
```

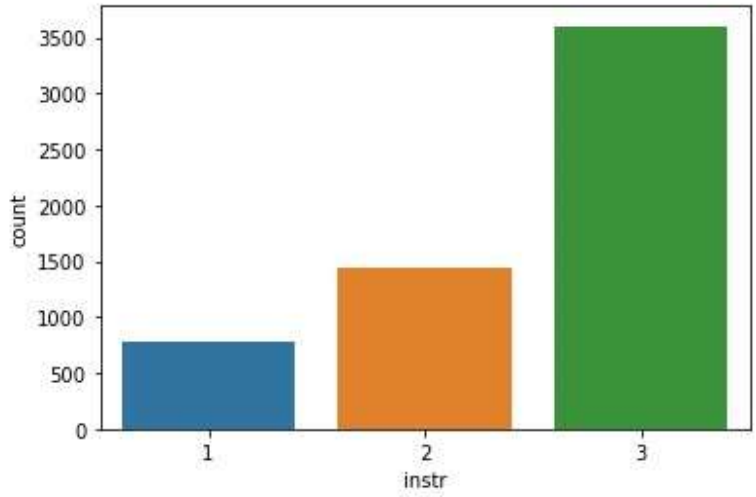
Out[6]:

	instr	class	nb.repeat	attendance	difficulty	Q1	Q2	Q3	Q4	
count	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.0
mean	2.485567	7.276289	1.214089	1.675601	2.783505	2.929897	3.073883	3.178694	3.082474	3.1
std	0.718473	3.688175	0.532376	1.474975	1.348987	1.341077	1.285251	1.253567	1.284594	1.2
min	1.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0
25%	2.000000	4.000000	1.000000	0.000000	1.000000	2.000000	2.000000	2.000000	2.000000	2.0
50%	3.000000	7.000000	1.000000	1.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.0
75%	3.000000	10.000000	1.000000	3.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.0
max	3.000000	13.000000	3.000000	4.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.0

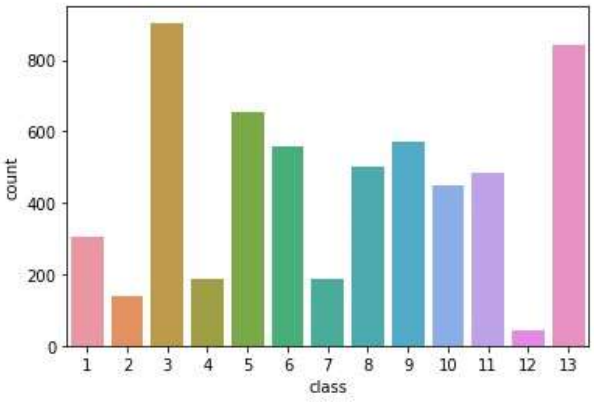
O 2.3) Exploratory Data Analysis

- Instruction three has taken more courses
- Maximum number of class

Out[14]: <AxesSubplot:xlabel='instr', ylabel='count'>



Out[19]: <AxesSubplot:xlabel='class', ylabel='count'>



.4) Cor

Relation Matrix

- Use to Find Co-relation between columns.
- Get useful columns from the data.

In [31]:
```
corr=df.corr()
plt.figure(figsize=(18,18))
sns.heatmap(corr,annot=True,cmap="coolwarm")
```

Out[31]: <AxesSubplot:>

o 2.5) Principal of Component Analysis

- To reduce the dimension of the data.

In [79]:
```
X=df.iloc[:,5:33]
from sklearn.decomposition import PCA
pca=PCA(n_components=2,random_state=42)
X_pca=pca.fit_transform(X)
```

In [80]:
```
X_pca
# So now dimensions are reduced to Two
```

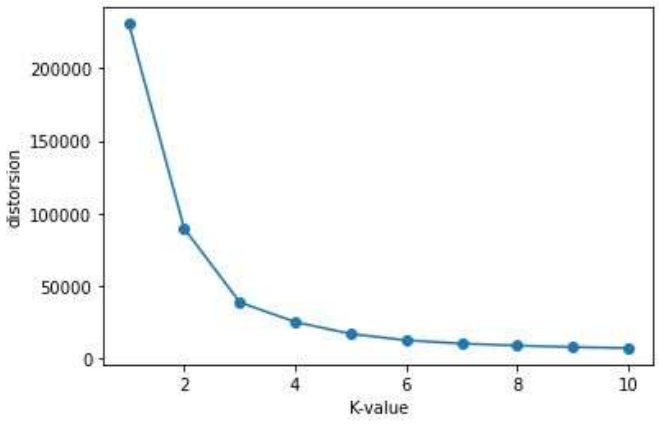
Out[80]:
```
array([[ 0.98901533,  0.52279815],
       [ 0.98901533,  0.52279815],
       [-9.59128851,  0.6408021 ],
       ...,
       [-9.59128851,  0.6408021 ],
       [11.56931918,  0.40479421],
       [11.56931918,  0.40479421]])
```

O 2.6) Model Building

- Making elbow to find the best value of K

In [85]:
```
from sklearn.cluster import KMeans
distortions=[]
for i in range(1,11):
    km=KMeans(n_clusters=i,init='k-means++',n_init=10,max_iter=300,random_state=0)
    km.fit(X_pca)
    distortions.append(km.inertia_)
plt.plot(range(1,11),distortions,marker='o')
plt.xlabel("K-value")
plt.ylabel("distorsion")
```

Out[85]:
```
Text(0, 0.5, 'distorsion')
```



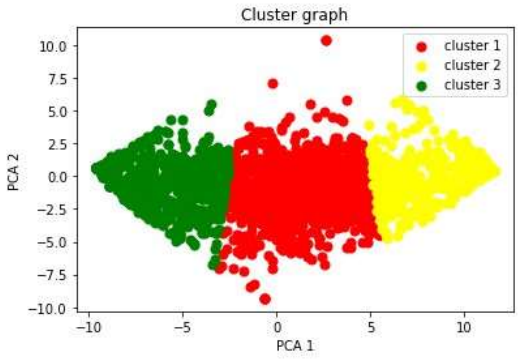
- Train the model k=3

```
In [87]: model=KMeans(n_clusters=3,init='k-means++',n_init=10,max_iter=300,random_state=0)
         model.fit(X_pca)

Out[87]: KMeans(n_clusters=3, random_state=0)

In [88]: y=model.predict(X_pca)

In [89]: y

Out[89]: array([0, 0, 2, ..., 2, 1, 1])
```

- Plot Graph to show cluster

Plot the Graph to show the Clusters

```
In [95]: plt.scatter(X_pca[y==0,0],X_pca[y==0,1],s=50,c="red",label="cluster 1")
         plt.scatter(X_pca[y==1,0],X_pca[y==1,1],s=50,c="yellow",label="cluster 2")
         plt.scatter(X_pca[y==2,0],X_pca[y==2,1],s=50,c="green",label="cluster 3")
         plt.title("Cluster graph")
         plt.xlabel("PCA 1")
         plt.ylabel("PCA 2")
         plt.legend()

Out[95]: <matplotlib.legend.Legend at 0x23bdef40e50>
```



- Agglomerative clustering

In [102…

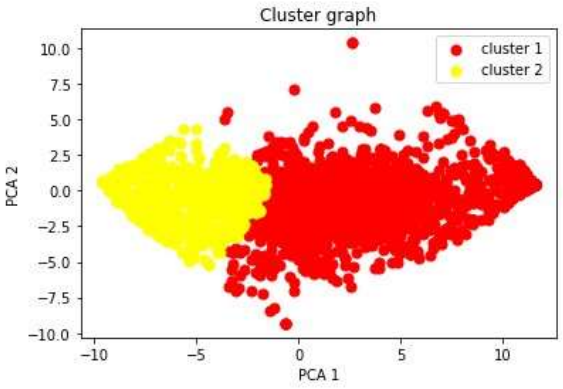
```
from sklearn.cluster import AgglomerativeClustering
model=AgglomerativeClustering(n_clusters=2,affinity='euclidean',linkage='ward')
y=model.fit_predict(X_pca)
```

In [103…

```
plt.scatter(X_pca[y==0,0],X_pca[y==0,1],s=50,c="red",label="cluster 1")
plt.scatter(X_pca[y==1,0],X_pca[y==1,1],s=50,c="yellow",label="cluster 2")
plt.title("Cluster graph")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.legend()
```

Out[103…

```
<matplotlib.legend.Legend at 0x23be1c8b3a0>
```

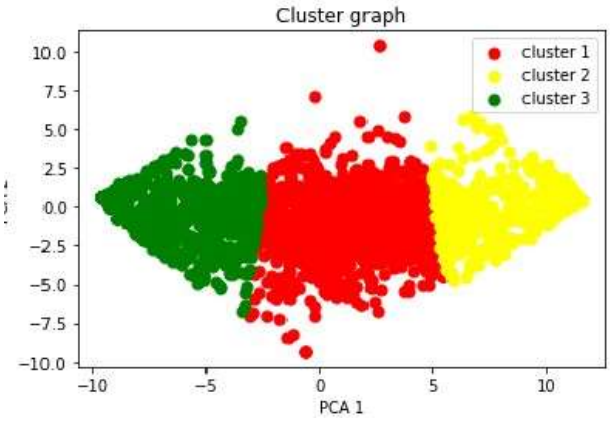


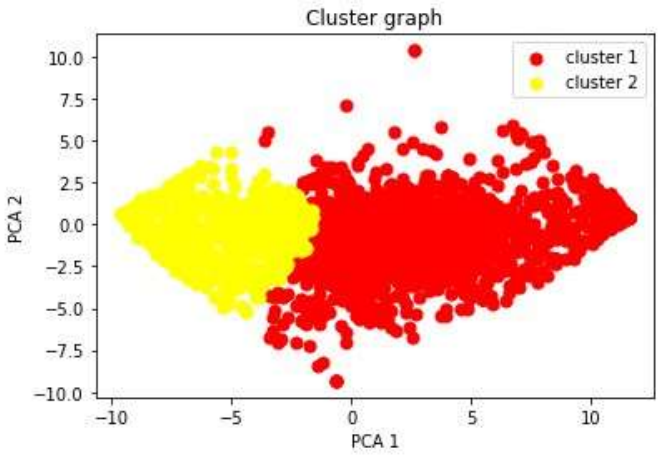
3) Result:-

1) We perform classification on Turkiye Student Evaluation Data.
2) Model:
   - ○ KMeans clustering: k=3 ○
   - Agglomerative clustering: k=2
3) Both the Model are predicting and showing great result.

4) Conclusion:

- We train our model on KMeans and Agglomerative.
- Now we divide the data into clusters.





5) **Reference(if any):**

- http://archive.ics.uci.edu/ml/datasets/turkiye+student+evaluation