Machine Learning.

Date:
Page No.:

Day 1. Introduction to Machine Learning.

- Supervised Learning
  - Linear regression, logistic regression, ridge and lasso's regression, Decision ~~Making~~ Tree algorithm, Random Forest Algorithm K-nearest neighbour., Support Vector Machine, Naive Bayes, boosting ....

- Unsupervised learning.
  - kmeans, PCA, Clustering, DBScan.

- Deep Learning -
  - Time series
  - Natural Langauge Processing (NLP)
  - Neural Networks (ANN, CNN, etc)

Day 2.

* Numpy Library. :- Numerical Python.

1) It is a Library consisting of multidimensional array objects and a collection of routines for processing those array.

Q Identity Matrix.
→ diagonal matrix will be one and rest everything will be zero.

Date:
Page No.:

Day 3. Numpy Part 2.

1) Indexing And slicing.

eg:- a = np.array([[1,2,3],[4,5,6]])
a
a[0,0] # 1st Row, 2nd Column.

2) Slicing of 2-D array.

eg:- a = np.array([[1,2,3],[4,5,6],[7,8,9]])
a
a[1:2, 1:2] # 1st Row, 2nd Column.

3) Append.

eg:- a = np.array([[1,2,3],[3,4]])
a
b = np.append(a, [[5,6]], axis=0)
b.

4) np.mean(a)
5) np.median(a)
6) np.std(a).

Day 4 Pandas.

1) Used for data Manipulation and data Analysis.
2) Pandas is baild on the top of the numpy package
3) Panel Data ----> Pandas
4) In Pandas we have 2 data structure ----->
   i) Series ii) DataFrame.

Date:
Page No.:

Series
1) it has a sequence of data values
eg:- pd.Series([1,2,3,4,5])

DataFrame
1) A DataFrame is nothing but a table.
eg:- pd.DataFrame({'Yes':[50,21,56], 'No':[101,20,11]})

eg:- pwd() # For File path

eg:- df = pd.read_csv(r"Data.csv").
df
df.columns
df.shape
df.head()
df.tail()
df.info()
df.isnull()
df.isnull().sum().

DAY 5 Pandas Part 2.

Drop.
eg:- df.drop('e', axis=1, inplace=True)
df

#
df['math'].max()
min()
mean()
median()

Date:
Page No.:

Value_count.

eg:- df.name.value_counts().

unique & nunique.

df['result'].nunique() #count.
df['result'].unique() # exact value.

From numpy.random import randn as rn.

rn(5,3).

Seed

np.random.seed(1)
r = rn(5,3)
r

rdf = pd.DataFrame(r)
rdf.

loc() & iloc()

df.loc[[2,4],['name','result']] # Values
→ select by label

df.iloc[[2,4],[0,3]] # index.
→ select by index or position.

rdf.iloc[0:3,0:2]

Date:
Page No.:

```
rdf[rdf > .5].
```

Fillna()

```
df = df.Fillna(5).
df,
df = df.Fillna ('good')
```

Concat()

```
df1 = pd.DataFrame({'A':['A0' A1 A2], 'B':['B0', B[,B]
,'C':['C0','C1','C2']}).
df1
df2 = pd.DataFrame({'A':[A3 A4 A5], 'B':[B3 B4 B]
,'C':[C3 C4 C5]}).
```

Day 6. Pandas.

Merging.

eg:
```
left = pd.DataFrame({'key':['K0','K1','K2'],
                     'A':['A1','A2','A3'],
                     'B':['B1','B2','B3']}).
left.

right = pd.DataFrame({'key':['K0','K1','K2'],
                      'C':['C1','C2','C3'],
                      'D':['D1','D2','D3']}).
right
```

Date:
Page No.:

eg: left

eg: right.

eg: inner = pd.merge (left, right, how='inner', on='key')
inner

eg: leftjoin = pd.merge (left, right, how='left', on='key')
leftjoin.

eg:- rightjoin = pd.merge (left, right, how='right', on='key')
rightjoin.

groupby

eg:- data = {'Company': ['GOOG', 'GOOG', 'MSFT', 'MSFT', 'FB', 'FB'],
'Person': ['SAM', 'CHARLIE', 'AMY', 'VENSSA', 'CARL', 'SARAH'],
'SALES': [200, 120, 360, 124, 243, 350]}
df = pd.DataFrame (data)
df

eg:- bycomp = df.groupby ('Company')
bycomp

eg:- bycomp.mean()
bycomp.median()
bycomp.std()

Sorting

eg:- df. = pd.DataFrame (age:list, columns = ['Country', 'Year', 'Population',
'Continent'])
df

eg df.sort_values (by = ['Country'])

eg df.sort_values (by = ['Population'], ascending = False).

eg: df.sort_index (axis = 1)

Banking CreditRisk Analysis.

Date:

Page No.:

Day 7. Data cleaning.

```
eg:- import numpy as np.
     import panday as pd.
     import matplotlib.pyplot as plt.
     import seaborn as sns

eg: cr.pd.read_csv(r"CreditRisk.csv")
    cr
eg: cr.head(2).
eg: cr.shape
eg: cr.info()
eg: cr.isnull().sum()
eg: cr.Education.value_counts()/cr.shape[0] * 100
    # For percentage
eg  cr.Education.unique().
eg  cr.Self_Employed.value_counts()
eg  cr.Self_Employed = cr.Self_Employed.Fillna('no')
eg: cr.isnull().sum()
eg  cr.self-Employed = cr.Self_Employed.replace({'no':'No'})
eg:- cr['New-App_inc'] = 2* cr.ApplicantIncome
eg:-cr.shape
eg:- cr.head(2)
eg:- cr.LoanAmount.value_counts()
eg:- cr.LoanAmount = cr.LoanAmount.Fillna(cr.loanAmount
                                  mean())
    # Replace by mean.
eg:- cr.isnull().sum().
eg:- cr.iloc[[1,100,500,600,800],[1,6]].
```

penkraft

Date:
Page No.:

Day 8. What is Sampling And Why we do it ?.

i) Sampling :- Sampling is the process of selecting a subset of data from a large dataset to use in ML model.

ii) Studies with inadequate sample suffer from overfitting of data and have a lower probability of producing true effects.

iii) The increment in sample size increases the accuracy of prediction but may not cause a significant change after a certain sample size.

There are main types of sampling techniques

i) Random Sampling

ii) Stratified Sampling.

i) Random Sampling.

in which each item in the population has an equal chance of being selected in the sample.

No'1)

In sampling we will divide it into ~~data~~ in ; train data and test data.

☆ Because on train data built the model on test data test the model

sunt

More data in train & less data in test.

eg:- placement training should be more (more knowledge) then we will get the job.

Date:
Page No.:

* The ratio of the train : test is ?
  70% : 30%

Why Random.

→ Random sampling to avoid the biasness.

Why train & test ?

→ Testing should not be done on same data which you have building the model.

eg:- Teaching — Training.
Exam — Testing.

if you do the testing on the same data on which you have builded the model, you will get the accuracy which very high but it will be a high false accuracy.

Day 9. Steps involved in Model Building.

1. Understanding the problem statement or requirement.
2. Get the data / Local the data.
3. Data pre-processing / ~~Local the~~ Data cleaning
4. Sampling (Random sampling i.e. Dividing in train and test).
5. Building the model.
6. Test the model and check the performance
7. Re run the model from some previous step

Date:
Page No.:

step1. Reading the Data.
step2. Data Cleaning / Data preprocessing.

Note :- if there in the column there is more than 60 % null values we want to drop the table.

Step3 Sampling.
eg:- From sklearn.model_selection import train_test_sp
eg:- cr_train, cr_test = train_test_split(cr, test_size = .2)
eg:- cr_train.shape
eg 981 * 0.8
eg cr_test.shape
eg. 981 * .2

Day 10 Introduction to Linear Regression

Machine Learning :-
1. Supervised Learning
2. Unsupervised Learning.

Properties of Supervised Learning.-
1. Target variable is present.
2. Sampling is possible in Supervised Learning

Properties of Unsupervised Learning
1. Target variable is not present.
2. Sampling is not possible in Unsupervised Learning.

Date:
Page No.:

1) The machine that learn to that input new data.

✱ Supervised Learning.

2) The machine will be train on know data. or label data. it will predict data.

3) Supervised learning often use for classification problem, Regression problem and object dectection problem.

✱ Unsupervised.

known Data Model response.

The unsupervised ↓ unlabel Data while train in the model it will create the patterns.

Model will Findout the pattern is known as clustering

The unsupervised learning is often use to perform the task as clustering, diamentionality Reduction.

1) Supervised Learning

It was classified into two categories of algorithm.

Date:
Page No.:

1. Regression -

A regression problem is when the output variable is a real and continuous value.

eg:- weight & height (Integer).

2. Classification -

A classification problem is when the output variable is a category

eg:- Grender, Colour (Object).

1. Regression :-

A. Linear Regression - "Straight Line" to do the prediction.

(∵ Supervised).

* Equation of Linear Regression :-

$$Y = B_0 + B_1 x_1 + B_2 x_2 + \cdots + B_n x_n + E$$

(↑ error terms.)

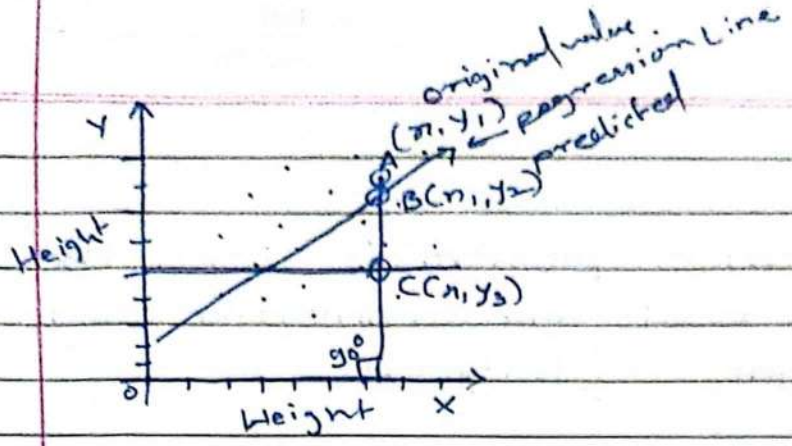
Consider I have a data.

X	Y
Weight	Mileage
1000	15
1200	13
1100	13.5
1300	11.9

Here; Y depends on X

Y is function X.

Any value falls on regression line should be predicted value.



Error = Actual - Predicted

Day 11. Linear Regression Part 1.

Actual	Predicted	Error	Sum sq of error (SSE)
40	43	-3	9
70	68	2	4
80	79	1	1
75	72	3	9
47	50	-3	9
		Sum = 0	32.

Sq of (Actual (mean) - Pred)

Mean (Actual)	Predicted	error	~~Sum sq error~~
62.4	43	19.4	376.36
62.4	68	-5.6	31.36
62.4	79	-16.6	275.56
62.4	72	-9.6	92.16
62.4	50	12.4	153.76
			185.94 (SSR)

Sum sq regression

Sum square regression is nothing but the mean value of sq. of (mean - Pred)

Date:
Page No.:

Predictive Modeling :-

* Evaluate the Model.
* No model is 100% accurate.
* if it is accurate it has overfitting or done any mistake.
* Evaluatting parameter & matrix.

Day 12 Linear Regression Part 2.

* Linear Regression is Predictive Modelling
* Evaluating parameter

For model evaluation,

1. I will use the matrix known as Rsquare or R2.

$R^2 = SSR / SST$

$= SSR / SSR + SSE$ .... Highly imp.

- R^2 range is between 0 to 1.
- R^2 will always be a positive number.
- Higher the R^2 will better the model.
  - if it is near to 1 is better
  - if it is near to 0 is not better.
- Choose the model which has higher R^2 value.

Q What is the problem associated with R^2 ?

→ We consider that higher the R^2 better the model but at a same time you keep on adding insignificant X variables and R^2 keep on increasing.

penkraft

Date:
Page No.:

MPG(Y) Wt of Car Cyl Fuel traffic colour interior

Model 1 - Consider only Wt of the car
$R^2 = 0.6$.

Model 2 - Consider Wt of the car + Cyl + Fuel
$R^2 = > 0.6$.

Model 3. - Consider Wt of car + Cyl + Fuel + traffic + color + interior
$R^2 >$.

✱ to solve the problem of R^2

2. Adjusted R^2 -

$$\text{Adj } R^2 = 1 - \left[\frac{(1-R^2)\times(N-1)}{N-K-1}\right]$$

N = No of Record or Rows
K = No of column or x variable.

- it will just give correct answer
- Adjusted R^2 can never be greater than R^2
- 0 to 1

penkraft

Date:
Page No.:

- Range of adj R^2 is -ve to 1
- If R^2 is moving towards 0 then Adj R^2 is moving towards -ve.
- then we have to recheck the model.
- Higher the adj R^2 better is the model

use of MSE Mean Square Error.
MSE use for the comparision

3. MSE (Mean Square Error) (Cost Function) also called as.

Advantage of using MSE.

1. Negative & positive error are getting cancel out.
2. MSE is used to punish the model for larger errors.
3. lower the MSE better is your model

MSE range will be 0 to infinity

Day 13 Linear Regression Part 3.

- MSE depends upon the range and magnitude of target variable.

4. RMSE (Root mean square error) -

RMSE = sq root MSE.

Date:
Page No.:

✱ Why we need RMSE ???

→

Actual	Pred	Error	MSE	RMSE
7 rupee	14 rup	7 rupp	49	7.

- The value will be shoot up.

5. ~~MAPE (Mean Abso~~

- RMSE gives ~~in~~ same answer in actual ~~unit~~ value are there while MSE gives answer in square unit

- Some time interpreting MSE would be ~~problem~~ problematic difficult because no. are getting shootup that why we RMSE.
- Lower the RMSE better is the model.

5) MAPE (Mean absolute percentage Error)

here,

$$\% \text{ error} = \frac{\text{error}}{\text{Actual}} \times 100$$

- Lower the MAPE better is the model

7%

Accuracy = 100 − 7 = 93%.

Assumptions in Linear Regression.

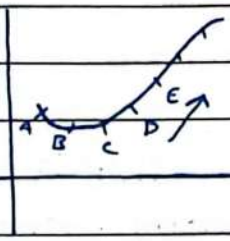
Assumption may be true or may not be true

imp

Date:
Page No.:

1. There is a linear relationship between X & Y
2. Error term are normally distributed and there is no pattern among them.
3. Minimum multicolinearity among X variable
4. Homosedacity - variance around the regression line is same for all the predicted values.

Q Why we assume it to be linear only?



— Dicision making should be difficult.

—

Date:
Page No.:

Day 14. Assumption in Linear Regression.

3. Multicolinearity :-
There should be high correlation between x and y variable.

Production of Factory (Y)	Machine x_1	Raw material x_2	Man power x_3	electricity x_5	total Labours x_6

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n + \epsilon$$

2. Error terms are normally distributed and there is no pattern among them.

4. Homosedacity - variance around the regression line should be same for all the predicted values.

* While building the model remove outliers to maintane the accuracy of the model.

** First Model on linear Regression.

Mandatory conditions for Linear Regression.

1. Target variable should always be numeric and continuous.
2. Number of rows (N) should always be greater than no. of columns (k).

Date:
Page No.:

eg:- lcn = pd.read_csv(r"LungCapData.csv")
lcn.

Requirement is to built the predictive model which can predict the Lung capacity.

3 step :- Data Cleaning.

eg:- lcn.shape
eg:- lcn.info()
eg:- lcn.isnull().sum()
eg:- lcn.describe()
eg:- lcn.Gender.value_counts()
eg:- lcn.Gender = lcn.Gender.replace({'male':1,'Female':0})
eg:- lcn
eg:- lcn.Smoke.value_counts()
eg:- lcn.Smoke = lcn.Smoke.replace({'yes':1,'no':0})
eg:- lcn
eg:- lcn.Caesarean.value_counts()
eg:- lcn.Caesarean = lcn.Caesarean.replace({'yes':1,'no':0})
eg:- lcn.
eg:- lcn.info()

Day 15 Linear regression model on lcn data.

Step 4 :- Sampling.

eg:- from sklearn.model_selection import train_test_split
eg:- lcn_train, lcn_test = train_test_split(lcn, test_size = .2)
eg:- lcn_train.shape # it will select Randomly data 580.
eg:- lcn_test.shape #

Date:
Page No.:

eg:- lcn_train_x = lcn_train.iloc[:, 1::]
eg:- lcn_train_y = lcn_train.iloc[:, 0]

eg:- lcn_train_x
eg:- lcn_train_y
eg:- lcn_test_x = lcn_test.iloc[:, 1::]
eg:- lcn_test_y = lcn_test.iloc[:, 0]
eg:- lcn_test_x
eg:- lcn_test_y

Step5 :- Building the model
#- Import the necessary Function From sklearn
#- Create an object of that Function
#- Run the FIT Function and then model is creat
#- Run the prediction (predict) the Funtion

eg:- From sklearn.linear_model import LinearRegression
eg:- linereg = LinearRegression()
eg:- linereg.fit(lcn_train_x, lcn_train_y)
eg:- pred_train = linereg.predict(lcn_train_x)
eg:- pred_test = linereg.predict(lcn_test_x)
eg:- pred_train.shape
eg:- pred_test.shape
eg:- error_test = lcn_test_y - pred_test
eg:- error_test

Step 6 :- Evaluate the model
R^2
eg:- Rsquare = linereg.score(lcn_train_x, lcn_train_y)
eg:- Rsquare

penkraft

Date:
Page No.:

eg:- N = lcn_train_x.shape[0]
eg:- k = lcn_train_x.shape[1].

Adj-Rsquare.
eg:- adj_Rsquare = 1-(1-Rsquare)*(N-1)/(N-k-1)
eg:- adj_Rsquare.

MSE.
eg:- mse = np.mean(np.square(error_test))
eg:- mse

RMSE.
eg:- rmse = np.sqrt(mse).
eg:- rmse

MAPE
eg:- mape = np.mean(np.abs(error_test*100/lcn_test_y)) (Actual value)
mape.

Day 16.

eg:- import pandas as pd.
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import numpy as np.

eg:- ppt = pd.read_csv(r"Property_Price_Train.csv")
ppt.
eg:- ppt.head()

Date:
Page No.:

Data Cleaning.

eg:- ppt.shape.
eg:- ppt.info()

eg:- ppt.isnull().sum()

eg:- len(ppt.isnull().sum()[ppt.isnull().sum()>0])
eg:- ppt.isnull().sum()[ppt.isnull().sum()*100/ppt.shape[0]>40]

eg:- ppt.Lot_Extent.value_counts().
eg:- ppt.LotExtent = ppt.LotExtent.Fillna(ppt.Lot_Extent.mean())
ppt.Lot_Extent
eg:- ppt.Brick_Veneer_Type.value_counts().
ppt.Brick_Veneer_Type = ppt.Brick_Veneer_Type.Fillna('Non

eg:- ppt = ppt.drop(['Pool_Quality', ' ' ... ' '
], axis=1)
eg:- ppt.shape.
eg:- len(ppt.isnull().sum()[ppt.isnull().sum()>0])
eg:- ppt = ppt.iloc[:, 1:]
eg:- ppt.head()
eg:- ppt.shape

Date:
Page No.:

imp

Label Encoder.

eg:- From sklearn.preprocessing import LabelEncoder
eg:- le = LabelEncoder()

eg:- ppt[ppt.select_dtypes(include=['object']).columns]
= ppt[ppt.select_dtypes(include=['object']).columns]
.apply(le.fit_transform)

eg:- ppt.head()

eg:- ppt.info()

Sampling

eg:- from sklearn.model_selection import train_test_split
eg:- ppt_train, ppt_test = train_test_split(ppt, test_size=0.25)

eg:- ppt_train_x = ppt_train.iloc[:, 0:-1]
ppt_train_y = ppt_train.iloc[:, -1]

eg:- ppt_train_x

eg:- ppt_test_x = ppt_test.iloc[:, 0:-1]
ppt_test_y = ppt_test.iloc[:, -1]

eg:- ppt_test_x

Date:
Page No.:

Day 17. Checking Assumptions on the PPT data.

Building the model.

eg:- From sklearn.linear_model import LinearRegression
linereg = LinearRegression()

eg:- linereg.fit(ppt_train_x, ppt_train_y)

eg:- pred_train = linereg.predict(ppt_train_x)
pred_test = linereg.predict(ppt_test_x)

eg:- err_test = ppt_test_y - pred_test.
err_test.

eg:- Rsquare = linereg.score(ppt_train_x, ppt_train_y)
Rsquare

eg:- N = ppt_train_x.shape[0]
k = ppt_train_x.shape[1]

eg:- adj_Rsquare = 1 - (1 - Rsquare)*(N-1) / (N-k-1)
adj_Rsquare

eg:- linereg.coef_ # β values.

eg:- linereg.intercept_ # β_0 value.

eg:- mse = np.mean(np.square(err_test))
mse

penkraft

Date:
Page No.:

eg:- rmse. = np.sqrt(mse)
rmse.

eg:- mape = np.mean(np.abs(err_test * 100/ppt_tst_y)
mape.

Lets check the assumptions
Assumption on the train data.

eg. err_train = ppt_trainy - pred_train
err_train

Error terms are normally distributed

eg:- import matplotlib.pyplot as plt.
eg:- plt.plot(err_train, '*')

eg:- plt.hist(err_train, bins = 70, edgecolor = 'r')
this is not a normally distributed graph.

eg. pred_actual_df = pd.DataFrame()

eg:- pred_actual_df['Actual'] = ppt_train_y
pred_actual_df['Predicted'] = ppt_pred_train
eg:- pred_actual_df

eg import seaborn as sns

eg sns.jointplot(x = 'Actual', y = 'Predicted', data = pred_actual_df
, kind = 'reg').

Date:
Page No.:

Day 18. LCN data and Homework.

Continue From Day 15 ....

eg:- err_train = lcn_train_y - pred_train
err_train

Error are normally distributed or not ?

eg:- import matplotlib.pyplot as plt.
eg:- plt.plot (err_train, '.')

eg:- plt.hist (err_train, bins=30, edgecolor='g');

eg:- new_data = pd.DataFrame()

eg:- new_data['Actual'] = lcn_train_y
new_data['Predicted'] = pred_train

eg:- new_data

eg:- import seaborn as sns

eg:- sns.jointplot (x='Actual', y='Predicted', data=new_data
, kind='reg').

Date:
Page No.:

Day 19 Car price data and Removing Outliers.

eg:- import pandas as pd.
import numpy as np.

eg:- car = pd.read_csv(r"car_price.csv")
car

eg:- car.shape.

eg:- car.head()

We have special characters in our data
We will convert this special characters top null values
Once we convert into nulls then we already how to handle the nulls.

eg: car = car.replace({'?': np.nan})

eg: car.head()

eg: car.isnull().sum()

eg: car.isnull().sum()[car.isnull().sum()>0]

eg: car.normalized_losses.value_counts()

eg:- car.normalized_losses = car.normalized_losses.astype('Float')
car.bore = car.bore.astype('float')
car.stroke = car.stroke.astype('Float')
car.horsepower =
car.peak_rpm =
car.price. =

eg:- car.normalized_losses = car.normalized_losses.fillna(car.normalized_losses.mean())

eg:- car.bore = car.bore.fillna(car.bore.mean())

Date:
Page No.:

car.stroke =
car.horsepower =
car.peak_rpm =

eg:- car.isnull().sum()

eg:- car.num_of_door.value_counts()
eg:- car.num_of_door = car.num_of_door.fillna('Four')
eg:- car = car.dropna() # it will drop the rows
eg car.shape

eg:- From sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

eg:- car[car.select_dtypes(include=['object']).columns] =
car[car.select_dtypes(include=['object']).columns]
.apply(le.fit_transform)

eg:- car.info()
eg:- car.head()

eg:- From sklearn.model_selection import train_test_split
eg:- car = df1 # store in df1 for remove outliers
eg:- car_train, car_test = train_test_split(car, test_size=0.2)
eg:- car_train_x = car_train.iloc[:,:-1]
car_train_y = car_train.iloc[:,-1]
eg car_train_x
car_train_y
eg:- car_test_x = car_test.iloc[:,0:-1]
car_test_y = car_test.iloc[:,-1]
eg car_test_x

penkraft

Date:
Page No.:

eg:- From sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(car_train_x, car_train_y)

eg:- pred_train = lr.predict(car_train_x)
pred_test = lr.predict(car_test_x)

eg:- err_test = car_test_y - pred_test
err_test

eg:- Rsquare = lr.score(car_train_x, car_train_y)
Rsquare

eg:- N = car_train_x.shape[0]
K = car_train_x.shape[1]

eg:- adj_Rsquare = 1-(1-Rsquare)*(N-1)/(N-K-1)}
adj_Rsquare

eg:- mse = np.mean(np.square(err_test))
mse

eg:- Rmse = np.sqrt(mse)
Rmse

eg:- mape = np.mean(np.abs(err_test*100/car_test_y))
mape

eg:- err_train = car_train_y - pred_train
err_train

eg:- import matplotlib.pyplot as plt.

eg:- plt.plot(err_test, '.')

eg:- plt.hist(err_train, bins=30, edgecolor='r');

Date:
Page No.:

eg:- preal_actual_df = pd.DataFrame()

eg:- preal_actual_df['Actual'] = car_train_y
preal_actual_df['Predicted'] = preal_train

eg:- preal_actual_df

eg:- import seaborn as sns

eg:- sns.jointplot(x='Actual', y='Predicted', data=preal_actual_df, kind='reg')

For removing Outliers

eg:- def remove_outliers(df, col, k):
    mean = df[col].mean()
    global df1
    sd = df[col].std()
    final_list=[x for x in df[col] if (x > mean-k*sd)]
    #upper Limit
    #lower Limit
    final_list=[x for x in df[col] if(x < mean+k*sd)]
    df1 = df.loc[df[col].isin(final_list)];
    print(df1.shape)
    print('Number of outliers removed ---->', df.shape[0] - df1.shape[0])

eg:- remove_outliers(car, 'price', 2)

pass x → [model] → get y

Date:
Page No.:

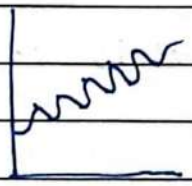
Day 20 Regularization Technique.

Regularization Technique -
- Ridge and Lasso's regression

Note:- There one problem in Linear regression is that there is high chance that model will get OverFit.

- OverFitting is ~~the~~ when the model will give very good results on train and gives bad results on test data.

In Overfitting the model will ~~memorize~~ try to mimic the data (i.e. mean it will adjust the all data point cover all the datapoints)



Lets say there are two points in my training data. And i am having two features or variables x & y

* OverFitting :-

Train Accuracy - 90 - 98 %.
Test Accuracy - 66 - 65 %

Low bias
High Variance

Date:
Page No.:

* Underfitting	Balanced fit / Generalized
Train Accuracy - 60%. Test Accuracy - 62%.	Train Accuracy - 90% Test Accuracy - 88%.
High bias High Variance.	Low bias Low Variance.

* Reason For Overfitting.

1. Data used for training is not cleaned properly and it contains noise in it.
2. The model has a high variance there will be chance get your model Overfit.
3. The size of training data used is very small.
4. The model is too complex.

* Ways to tackle of.
1. Regularization Technique
2. Cross - validation Technique.
3. Training your model with sufficient data.

* Theory of Regularization Technique

Q. How this ridge & lasso's avoid the overfitting

→ To prevent the issue of overfitting we are

Date:
Page No.:

going to use ridge regression

* - Ridge regression can also be called as L2 regularization Technique

MSE

$1/n * (X - Y)^2 = 1/n * 0 = 0.$

Ridge regression will add two values in this formula

$= (X - Y)^2 + \text{lambda} * (m)^2$ ← slope.

lambda slope square is just penalizing the equation so that we should not get zero accuracy.

* - Lasso's Regression (L1 regularization)

$= (X - Y)^2 + \text{lambda} |\text{slope}|$ ← absolute of slope.

↑ Constant

* For the Feature selection we use the lasso's regression.

- Feature selection

$Y = B_0 + B_1X_1 + B_2X_2 + \cdots$

Date:
Page No.:

Day 21 Ridge and Lasso's on Train Data.

```
eg:- import pandas as pd.
     pd.set_option ('display.max_columns', None)
     pd.set_option('display.max_rows', None)
     import numpy as np

eg:- tt = pd.read_csv (r"train.csv").
eg:- tt.head()
eg:- tt.= tt.drop(['ID'], axis=1)
eg:- tt.isnull().sum().
eg:- tt.shape.

eg:- From sklearn.preprocessing import LabelEncoder
eg:- le = LabelEncoder()
eg:- tt[tt.select_dtypes(include=['object']).columns] =
     tt[tt.select_dtypes(include=['object']).columns]
     .apply(le.fit_transform).
eg:- tt.head()
eg:- tt.info()

eg:- From sklearn.model_selection import train_test_split
eg:- tt,train, tt_test = train_testsplit(tt, test_size=.2)
eg:- tt_train_x = tt_train.iloc[:,1::]
     tt_train_y = tt_train.iloc[:,0]
eg:- tt_test_x = tt_test.iloc[:,1::]
     tt_test_y = tt_test.iloc[:,0].

eg:- From sklearn.linear_model import LinearRegression
eg:- linereg = LinearRegression()
eg:- linereg.fit(tt_trainx, tt_train_y)
```

Date:
Page No.:

eg:- Rsquare = linereg.score(tt_trainx, tt_trainy)
Rsquare

eg:- N = tt_train_x.shape[0]
K = tt_train_x.shape[1]

eg:- adj_Rsquare = 1-(1-Rsquare)*(N-1)/(N-K-1)
adj_Rsquare

eg:- linereg.coef_

eg:- linereg.intercept_

eg:- pred_train = linereg.predict(tt_train_x)
pred_test = linereg.predict(tt_test_x)

eg:- err_test = tt_test_y - pred_test
err_train = tt_train_y - pred_train

eg:- mse_test = np.mean(np.square(err_test))
mse_test

eg:- mse_train = np.mean(np.square(err_train))
mse_train

eg:- rmse_test = np.sqrt(mse_test)
rmse_test

eg:- rmse_train = np.sqrt(mse_train)
rmse_train

eg:- mape_test = np.mean(np.abs(err_test)*100/tt_test_y)
mape_test

eg:- mape_train = np.mean(np.abs(err_train)*100/tt_train_y)
mape_train

eg:- pred_test.max()

eg:- pred_test.min()

We are facing the problem of overfitting so avoid this problem we need to use the regularization technique.

penkraft

Date:
Page No.:

* Ridge Regression.

eg:- From sklearn.linear_model import Ridge
eg:- rid_reg = Ridge()
eg:- rid_reg.fit(tt_train_x, tt_train_y)
eg:- rid_reg.score(tt_train_x, tt_train_y)
eg:- pred_rid_train = rid_reg.predict(tt_train_x)
pred_rid_test = tt_test_x
eg:- err_rig_train = tt_train_y - pred_rid_train
eg:- mse_train_rid = np.mean(np.square(err_rig_train))
eg:- mse_train_rid
eg:- mse_test_rid
eg:- mape_train = np.mean(np.abs(err_train * 100 / tt_train_y))
mape_test =
eg:- mape_train
eg:- mape_test

* Lasso's Regression.

eg:- From sklearn.linear_model import Lasso
eg:- las_reg = Lasso()
eg:- las_reg.fit(tt_train_x, tt_train_y)
eg:- las_reg.score(tt_train_x, tt_train_y)
eg pred_train_las = las_reg.predict(tt_train_x)
pred_test_las = las_reg.predict(tt_test_x)
eg err_train_las = tt_train_y - pred_train_las
err_test_las = tt_test_y - pred_test_las
eg mse_train_las = np.mean(np.square(err_train_las))
mse_test_las = np.mean(np.square(err_test_las))

penkraft

Date:
Page No.:

Day22 Confusion Matrix.

1) - Confusion Matrix is a evaluation Matrix use for evaluating the classification algorithms.

2) Original or actual values - Y or N & 0 or 1
Predicted value. Y or N & 0 or 1

		Pred 0	Pred 1
Acted	0	22	4
	1	36	47

Q.

	0	1
0	9	4 x
1	5 x	7

* Accuracy = Total no of correct prediction / Total record.

$= \frac{11}{20} = 0.55 = 55\%$.

	0	1
0	5000	200
1	340	50

Model A.

	0	1
0	4000	1200
1	110	280

Model B.

Accuracy Model A = $\frac{5050}{5590} = 0.90 = 90\%$.

Accuracy Model b = $\frac{4280}{5590} = 0.76 = 76\%$.

* Accuracy cannot be only parameter to judge the model
* We only do not focus on Accuracy we also want to focus on classes

penkraft

Date:
Page No.:

Model B is good than Model A.

Actual \ Pred	0	1
0	5000 (TN)	200 (FP)
1	340 (FN)	50 (TP)

* Class 0 will also be called as Class N (Negative)
* Class 1 can also be called as P (Positive)

1) TPR (True Positive Ratio) (Recall)

$$= \frac{TP}{(TP + FN)} = \frac{50}{50 + 340} = \frac{50}{390} = 0.12$$

* TPR is always be greater

2) ~~False~~ FPR (False Positive Ratio)

$$\frac{FP}{(FP + TN)} = \frac{200}{200 + 5000} = \frac{200}{5200} = 0.0384.$$

* FPR to Find the error in class 0
* FPR is alway be lesser.

Date:
Page No.:

Day 23 Confusion Matrix part 2.

Q) How to choose the best model ?

- Choose a model which is balanced
balanced :- performace on both the model should be good.
- Choose the model which meets your problem statement or requirement.

		Pred.	
		Appro	Reject.
Actual	Appro	4200(TN)	150(FP)
	Reject	190(FN)	50(TP)

Model A

		Pred.	
		App	Rej
Actual	App	3200(TN)	750(FP)
	Rej	90 (FN)	450 (TP)

Model B.

Acc	92
TPr	20.83
FPr	3

Acc	81.8
TPr	83.3
FPr	18

	0	1
0	5000 (TN)	200 (FP)
1	340 (FN)	50 (TP)

	0	1
0	4600(TN)	300 (FP)
1	110 (FN)	280 (TP)

If you are working for a telecomm major (jio or airtel) and there is problem that many customers are leaving and you want to find those customers which are ~~close~~ likely to leave so that we can take the preventive action Class 0 not leaving and class 1 ppl leaving.

Acc	90.33	92
Tpr	12.82	71.71
Fpr	3	6

penkraft

Date:
Page No.:

4. Hit Ratio = $\frac{TP}{TP + FP}$
Precision

* Higher the precision the model is good.

5. AUROC CURVE.

Scenario 1 = You are getting 96% and u want 99%.
Scenario 2 = You are getting 50% and u want 75%.

Which scenario is easy to implement?

if TPR increase bcz TPR increase slowly after that FPR will not stop it will increase very fast.

Beyond a certain point when you tried to increase the TPR TPR will increase very slowly AND FPR increase very fast on the certain points Things will change ~~Destrictly~~ Drastically

6. F1 ratio
Harmonic mean bet pre and recall.
section a - 150 section b - 400
1/150 1/400
✓ ×

F1 ratio = 2* Precision * recall / Precision + Recall

High the F1 ratio better the model
range is like 0 to 1

penkraft

Date:
Page No.:

1. Accuracy - Higher
2. TPR - Higher
3. FPR - Lower
4. Precision - Higher
5. AUROC - Higher
6. FLRatio - Higher.

Day 24 Class Imbalanced

	0	1
0	7523(TN)	340(FP)
1	110 (FN)	30 (TP)

Loan Approval System
1000 Records
960 Rejected
40 Approved

Accuracy	94
TPR	21.822
FPR	4.3
Precision	8.1
FLration	11.7

Class imbbanced is representation of one class is much more than the other class, in such scenario your model do very good for the class which is having more representain & will perform bad on the class which has less representation.

In Class imbalanced overall accuracy but as we know that accuracy is not only parameter to judge the model.

Solution for class imbalanced.
There two solution -

penkraft

Date:
Page No.:

1. Over Sampling
2. Under Sampling

1. Over Sampling :- we increase record of the class which is under representate

2. Under Sampling we will decrease the record of the class which is over ~~reduce~~ represented.

— Oversampling or undersampling should be done only over the sampling

— Over & under sampling should be done only on train data

because in oversampling we make duplicate of the record which are under representate
⇒ if you ~~make~~ duplicate in test data is will give false accuracy that why so we are avoiding test data

Day 25 Logistic Regression.

* Logistic Regression.

1. Logistic is superviseed machine learning algorithm.
2. Target variable will be present
3. Sampling is possible.

Date:
Page No.:

2. Logistic regression target variable is always binary.
i.e. Yes or no, 0 and 1.

3. This Confusion matrix will be used to evaluate the logistic regression model.

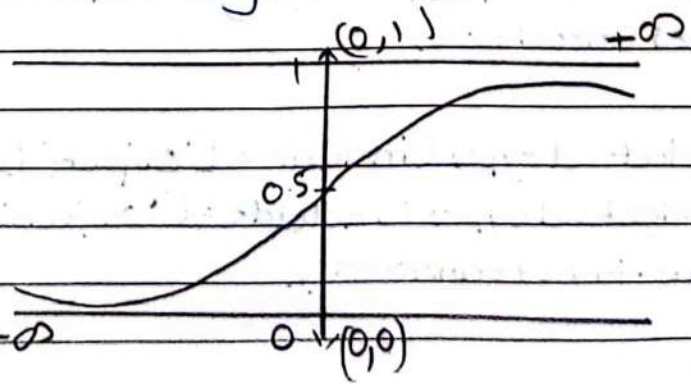
4. X variable can be anything

5. In logistic reg Y can take only 2 values ~~the~~ i.e. 0 or 1. & its work on principle of probability.
~~there~~ will be 2 probability.
Class 1 you will have one probability
Class 0 you will have one probability.

$p + q = 1$
$0.4 + q = 1$
0.6.

Whichever probability is more the Final

S curve or Sigmoid curved :-



Date:
Page No.:

eg:- import pandas as pd.
import numpy as np.
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

eg:- cr = pd.read_csv(r"CreditRisk.csv")
eg:- cr.head
eg:- cr.shape
eg:- cr.isnull().sum
eg:- cr.Gender = cr.Gender.Fillna('Male')
cr.Married = cr.Married.Fillna('No')
cr.Dependents = cr.Dependents.Fillna(0)
cr.Self_Employed = cr.Self_Employed.Fillna('Yes')
cr.LoanAmount = cr.LoanAmount.Fillna(cr.LoanAmount.mean())
cr.LoanAmount_Term = cr.Loan_Amount_Term.Fillna
(cr.Loan_Amount_Term.mean())
cr.Credit_History = cr.Credit_History.Fillna(0).

eg:- cr.isnull().sum()
eg:- cr = cr.drop(['Loan_ID'], axis=1)
eg:- cr.shape
eg:- cr.Loan_status = cr.Loan_status.replace({'Y':1,'N':0})
eg:- cr.info()

eg:- from sklearn.preprocessing import LabelEncoder.
le = LabelEncoder()

eg:- cr[cr.select_dtypes(include=['object']).columns] =
cr[cr.select_dtypes(include=['object']).columns].
apply(le.fit_transform).

Date:
Page No.:

eg:- cr.info()

eg:- From sklearn.model_selection import train_test_split
eg:- cr_train, cr_test = train_test_split(cr, test_size = .2)
eg:- cr_train_x = cr_train.iloc[:, 0:-1]
cr_train_y = cr_train.iloc[:, -1]
eg:- cr_test_x = cr_test.iloc[:, 0:-1]
cr_test_y = cr_test.iloc[:, -1]
eg:- cr_test_x
eg:- cr_test_y
eg:- cr_train_x
eg:- cr_train_y

eg:- From sklearn.linear_model import LogisticRegression
eg:- lr = LogisticRegression
eg:- lr.Fit(cr_train_x, cr_train_y)
eg:- pred = lr.predict(cr_train_x)
pred = lr.predict(cr_test_x)
eg:- pred
eg:- From sklearn.metrics import confusion_matrix
eg:- cr_confuse = confusion_matrix(cr_test_y, pred)
cr_confuse
eg:- cr_confuse.diagonal().sum() * 100 / cr_confuse.sum()
Accuracy
eg:- From sklearn.metrics import accuracy_score
eg:- accuracy_score(cr_test_y, pred) * 100

Date:
Page No.:

Day 26 Logistic regression model on CR data.

Continue ... (Day 25)

Recall
TP / TP + FN

eg:- From sklearn.metrics import recall_score
eg:- recall_score (cr_test_y, pred) * 100

Precission
TP / TP + FP

eg:- from sklearn.metrics import precision_score
eg:- precision_score (cr_test_y, pred) * 100

F1 score
2 * precision * recall / Precision + recall

eg:- from sklearn.metrics import f1_score
eg:- f1_score (cr_test_y, pred) * 100
eg:- pred = lr.predict (cr_test_x)
pred.
eg:- pred_proba = lr.predict_proba (cr_test_x)
pred_proba
eg:- from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
eg:- roc_auc_score (cr_test_y, pred) * 100
eg:- pred_proba [:, 1]
eg:- fpr, tpr, ther = roc_curve (cr_test_y, pred_proba [:, 1])
eg:- import matplotlib.pyplot as plt.
eg:- plt.plot (fpr, tpr)

Date:
Page No.:

eg:- plt.plot (Fpr, tpr, color = 'r')
plt.xlabel ('Fpr')
plt.ylabel ('Tpr')
plt.title ('Auroc on credit risk data')
plt.grid()
plt.text (x = .5, y = .5, s = 'Auroc is 73.59')

Day27. Logistic regression on Churn and Oversampling

eg:- import pandas as pd
import numpy as np
pd.set_option ('display.max_columns', None)
pd.set_option ('display.max_rows', None)

eg:- ch = pd.read_csv (r"Churn.csv")
eg:- ch.head()
eg:- ch.shape
eg:- ch.isnull().sum()
eg:- ch.info()
eg:- From sklearn.preprocessing import LabelEncoder
eg:- le = LabelEncoder()
eg:- ch[ch.select_dtypes (include = ['object']).columns] =
ch[ch.select_dtypes (include = ['object']).columns].
apply (le.fit_transform)
eg:- ch.info()
eg:- from sklearn.model_selection import train_test_split
eg:- ch_train, ch_test = train_test_split (ch, test_size = .2)
eg:- ch_train_x = ch_train.iloc [:, 0:-1]
ch_train_y = ch_train.iloc [:, -1]
eg:- ch_test_x = ch_test.iloc [:, 0:-1]
ch_test_y = ch_test.iloc [:, -1]

Date:
Page No.:

eg:- ch_train_x
eg:- ch_train_y
eg:- ch_test_x
eg:- ch_test_y
Overloading.
eg:- df1 = ch.train[ch_train_y == 1]
df1
eg:- df1.shape
eg:- ch_train = pd.concat([ch_train, df1, df1])
eg:- ch_train.Churn.value_counts()
eg:- ch_train.shape

eg:- From sklearn.linear_model import LogisticRegression
eg:- lr = LogisticRegression()
eg:- lr.fit(ch_trainx, ch_train_y)
eg:- pred = lr.predict(ch_train_x)
eg:- pred = lr.predict(ch_test_x)
eg:- pred

eg:- From sklearn.metrics import confusion_matrix
eg:- ch_confuse = confusion_matrix(ch_test_y, pred)
ch_confuse
eg:- ch_confuse.diagonal().sum() * 100/ch_confuse.sum()
eg:- from sklearn.metrics import accuracy_score
eg:- accuracy_score(ch_test_y, pred) * 100 # Accuracy
eg:- #Recall (TPR)
#TP/TP+FN
eg:- from sklearn.metrics import recall_score
eg:- recall_score(ch_test_y, pred) * 100

Date:
Page No.:

Precision
TP / TP + FP.

eg:- From sklearn.metrics import precision_score
precision_score (ch_test_y, pred) * 100

F1 score
2 * precision * recall / Precision + recall.

eg:- From sklearn.metrics import F1_score
F1_score (ch_test_y, pred) * 100.

eg:- pred = lr.predict (ch_test_x)
pred

eg:- pred_proba = lr.predict_proba (ch_test_x)
pred_proba

eg:- From sklearn.metrics import roc_auc_score
From sklearn.metrics import roc_curve

eg:- roc_auc_score (ch_test_y, pred) * 100

eg:- pred_proba [:, 1]

eg:- fpr, tpr, ther = roc_curve (ch_test_y, pred_pro
[:, 1])

eg:- import matplotlib.pyplot as plt.

eg:- plt.plot (fpr, tpr)

eg:- plt.plot (fpr, tpr, color = 'y')
plt.xlabel ('fpr')
plt.ylabel ('tpr')
plt.title ('Auroc on churn data')
plt.grid ()
plt.text (x = .5, y = .5, s = 'Auroc is 76.25')

Date:
Page No.:

Day 28 Model building on titanic data.

```
eg:- import pandas as pd
     import numpy as np.
     pd.set_option ('display.max_columns', None)
     pd.set_option('display.max_row', None)

eg:- Tt = pd.read_excel (r"titanic3.xls")
eg:- Tt.head()
eg:- Tt.shape
eg:- Tt.isnull().sum()
eg:- Columns_to_drop = Tt.columns[Tt.isnull().sum()*100
     / Tt.shape[0] > 40]
     print ('\n---- DROPED COLUMNS ---.\n', Columns_to_drop)
eg:- Tt. = Tt.drop(columns=['cabin', 'boat', 'body', 'home.dat',
     'name'])
eg:- Tt.shape
eg:- #Check unique values
     # Select only object columns
     object_columns = Tt.select_dtypes(include=['object'])

     #Check for non-numeric values in each object
     column & display unique symbols
     for column in object_columns:
         non_numeric_values = Tt[column].str.extract(r'[^\d]',
         , expand=False).dropna().unique()
         if len(non_numeric_values)>0:
             print(f"unique non-numeric symbols found in
                   column '{column}':")
             print(non_numeric values)
```

Date:
Page No.:

eg:- Tt.age = Tt.age.fillna(Tt.age.mean())
Tt.fare = Tt.fare.fillna(Tt.fare.mean())
Tt.embarkeal = Tt.embarkeal.fillna(Tt.embarkeal.mode()[0]

eg:- Tt.embarkeal.mode()
eg:- Tt.embarkeal.value_counts()
eg:- Tt.isnull().sum()

Preprocessing

eg:- from sklearn.preprocessing import labelEncoder
le = labelEncoder()
object_columns = Tt.select_dtypes(include=['object']).colu
Tt[object_columns] = Tt[object_columns].astype(str)
Tt[Tt.select_dtypes(include=['object']).columns] =
Tt[Tt.select_dtypes(include=['object']).columns].apply
(le.fit_transform).
eg:- Tt.corr()

# Rearrange the columns.
Tt = Tt[['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch'
, 'ticket', 'fare', 'embarkeal']]

Model Building.

eg:- from sklearn.model_selection import train_test_split
eg:- Tt_train, Tt_test = train_test_split(Tt, test_size=.3)
eg:- Tt_train.head()
eg:- Tt_test.head()

Date:
Page No.:

eg:- Tt_train_x = Tt_train.iloc[:,1::]
Tt_train_y = Tt_train.iloc[:,0]

Tt_test_x = Tt_test.iloc[:,1::]
Tt_test_y = Tt_test.iloc[:,0].

eg:- Tt_train_x
eg:- Tt_train_y
eg:- Tt_test_x
eg:- Tt_test_y

Building the model

eg:- From sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
eg:- lr.fit(Tt_train_x, Tt_train_y)
eg:- pred = lr.predict(Tt_test_x)
pred

Confusion Matrix.

eg:- From sklearn.metrics import confusion_matrix
eg:- Tt_tab = confusion_matrix(Tt_test_y, pred)
Tt_tab

Accuracy

eg:- From sklearn.metrics import accuracy_score
accuracy_score(Tt_test_y, pred)*100

Date:
Page No.:

Recall

eg:- from sklearn.metrics import recall_score
recall_score (Tt_test_y, preal) * 100

Precision

eg:- from sklearn.metric import precision_score
precision_score (Tt_test_y, preal) * 100

F1 ratio

eg:- from sklearn.metrics import f1_score
f1_score (Tt_test_y, preal) * 100.

eg:- preal_proba = lr.predict_proba (Tt_text_x)
preal_proba

Aucc

eg:- From sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
roc_auc_score (Tt_test_y, preal) * 100

Matplotlib.pyplot.

eg:- fpr, tpr, ther = roc_curve (Tt_test_y, preal_proba [:, 1])
import matplotlib.pyplot as plt
plt.plot (fpr, tpr).

Date:
Page No.:

Day 29 Decision Tree.

* Decision Tree can be built on binary class and Multi class also.

* Wherever we build the LR we can build the decision tree but the vise versa is not possible.

* Decision Tree has Target variable also

* if all the variable are not of same important then we need to check the most significant variable first to start the splitting of the tree

* In decision tree decision will be based on tree

* Whenever parameter or condition is most important that should be check first and then the second most go on.

- There are two Mathamatical formula

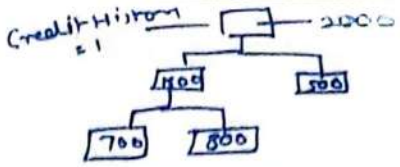
- How to decide parameter are most significant

- it will be decided based upon the mathematical formula (Entropy & gini)

- Either we Entropy or gini to decision tree but we can't used at both same time

Entropy } hyper parameter
Gini

penkraft



Date:
Page No.:

1. Entropy.

- Entropy is nothing but among of randomness in data
- Entropy tell us how homogeneous & hetrogenous sample is.
- if sample data is complete homogeneous information gain is 'zero'
- if sample data is complete hetrogeneous information gain is 'one'

Formula

$$\text{Entropy} = \Sigma - p \log_2^{(p)}$$

$$IG(Y, X) = E(Y) - E(Y|X)$$

(with respect to)

Ex:- Lets say I have 1000 records, 600 belongs to class 1 and 400 belongs to class 0 ?

prob for class 1 = 600/1000 ---> 6/10
prob for class 0 = 400/1000 ---> 4/10

Entropy of Y,

$$= -\frac{6}{10} \times \log_2\left(\frac{6}{10}\right) + \frac{4}{10} \times \log_2\left(\frac{4}{10}\right)$$

$$E(Y) = 0.97$$

Date:
Page No.:

560 CH is good, 440 CH is bad
560 Good CH, the loan is approved for 530 ppl and 30 ppl rejected
440 bad CH, the approved for 70 ppl and 370 loan rejected.

	0	1
0	370	70
1	30	530

$\frac{70}{440} = 15\%$

$E(Y|X) =$

1) loan status wis RSP Good CH.
→ $30/560 \log 30/560 + -530/560 \log 530/560$
$= 0.305$

2) loan status wis RSP bad CH
→ $-370/440 \log 370/440 + -70/440 \log 70/440.$
$= 0.624.$

For entire population
$= 560/1000 * 0.305 + 440/1000 * 0.624$
$= 0.1708 + 0.27456$
$E(Y|X) = 0.4453$

$IG(Y,X) = 0.97 - 0.4453$
$IG(Y,X) = 0.5247$

Date:
Page No.:

GINI Index or GINI Score.

Formula of GINI

$$1 - \Sigma P_i^2$$

Day 30 Decision Tree Part - 1.

Gini Index or Gini score

$1 - \Sigma P_i^2$

Loan status is our target variable, Calculating the Gini between Loan status and Credit History

Education	Credit History	Property Area	Loan status
Grad	1	Urban	N
Grad	1	Rural	N
Grad	0	Urban	Y
Grad	1	Urban	Y
Grad	1	Urban	Y
Non Grad	1	Urban	Y
Non Grad	1	Urban	Y
Grad	0	Semiurban	N
Non Grad	1	Urban	Y
Grad	0	Semiurban	Y
Grad	0	Semiurban	A.

CH == 1

True

7 record are satisfying the condition

False

4 record are not satisfying the condition

Date:
Page No.:

Here 5 record loan is approved
For 2 record loan is rejected

Here we are gonna use the Gini formula

$= 1 - [(5/7)^2 + (2/7)^2]$

$= 1 - \left[\frac{25}{49} + \frac{4}{49}\right]$

$1 - \frac{29}{49}$

$= 0.40$

Here, 2 records loan is app and 1 record loan is rej

$1 - [(2/4)^2 + (1/4)^2]$

$= 0.68.$

For entire population

$7/11 * 0.40 + 4/11 * 0.68$

0.392

Date:
Page No.:

Day 31 DT on Churn data and Hyper parameter.

eg:- from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
eg:- dtree.fit(BCCO_train_x, BCCO_train_Y)

eg:- dec_pred = dtree.predict(BCCO_test_x)
dec_pred

eg:- dtree_tab = confusion_matrix(BCCO_test_y, dec_pred)
dtree_tab.

* Hyper Parameter or Hyper parameter tunning.

Hyper Parameter are the parameter which are not part of data and user can control it or change it is called Hyperparameter.

→ depend on Criteria like model on Entropy or gini. (it is not part of Data) it depend on user.

D. In DT it our choice whether we want to build a tree or model using gini or entropy it is not a part of data but we can control it that why they are the hyper parameter.

hyperparameter are in infinite number of hyper parameter.

penkraft

Date:
Page No.:

Day 32 Decision on CTG. and Over sampling (Wine data.)

eg:-
```
import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
pd.set_option('display.max_row', None)
```

eg:-
```
W = pd.read_csv(r'wine.csv')
```
eg:-
```
W.head()
```
eg:-
```
W.shape()
```
eg:-
```
W.isnull().sum()
```

Sampling.

eg:-
```
from sklearn.model_selection import train_test_split
W_train, W_test = train_test_split(W, test_size=25)
```

eg:-
```
W_train_x = W_train.iloc[:,1::]
W_train_y = W_train.iloc[:,-1] # Selected by positions or index.

W_test_x = W_test.iloc[:,1::]
W_test_y = W_test.iloc[:,-1]
```

eg:-
```
W_train_x.head()
W_train_y.head()
W_test_x.head()
W_test_y.head
```

Decision Tree

Date:
Page No.:

eg:- From sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='gini', max_depth=None
dt.fit(W_train_x, W_train_y)

eg:- dec_pred = dt.predict(W_test_x)
dec_pred.

Confusion Matrix.

eg:- From sklearn.metrics import confusion_matrix.

eg:- dt_tab = confusion_matrix(W_test_y, dec_pred)
dt_tab

eg:- From sklearn.metrics import accuracy_score
accuracy_score(W_test_y, dec_pred) * 100.

Date:
Page No.:

Day 32 Feature selection using DT on CreditRisk

* Feature Selection

Higher the Feature importance value more imp or more significant is that variable or Feature

- Feature selection
- Lasso's regression is also used for the Feature selection.

eg:- import pandas as pd
import numpy as np.
pd.set_option('display.max_columns', None)
pd.set_option('display.max_row', None)

eg:- cr = pd.read_csv(r"CreditRisk.csv")
eg:- cr.head()
eg:- cr.shape
eg:- cr.isnull().sum()
eg:- cr.Gender = cr.Gender.Fillna('Male')
cr.Married = cr.Married.Fillna('No')
cr.Dependents = cr.Dependents.Fillna(0)
cr.Self_Employed = cr.Self_Employed.Fillna('Yes')
cr.LoanAmount = cr.LoanAmount.Fillna(cr.LoanAmount.mean())
cr.Loan_Amount_Term = ——h——
cr.Credit_History = cr.Credit_History.Fillna(0)

eg:- cr.isnull().sum()
eg:- cr = cr.drop(['Loan_ID'], axis = 1)
eg:- cr.shape

penkraft

Date:
Page No.:

Data Cleaning.

eg:- cr.loan_status = cr.loan_status.replace({'Y':1,'N':0}

eg:- cr.info()

eg:- From sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

eg:- cr[cr.select_dtypes(include=['object']).columns] =
cr[cr.select_dtypes(include=['object']).columns]
apply(le.fit_transform)

eg:- cr.info()

Sampling.

eg:- From sklearn.model_selection import train_test_split.
cr_train, cr_test = train_test_split(cr, test_size=.2)

eg:- cr_train_x = cr_train.iloc[:,0:-1]
cr_train_y = cr_train.iloc[:, -1]

cr_test_x = cr_test.iloc[:,0:-1]
cr_test_y = cr_test.iloc[:, -1]

Decision Tree.

eg:- From sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(cr_train_x, cr_train_y)

eg:- dec_pred = dtree.predict(cr_test_x)
dec_pred

penkraft

Date:
Page No.:

Confusion Matrix.

eg:- From sklearn.metrics import confusion_matrix

eg:- dtree_tab = confusion_matrix(cr_test_y, dec_pred)
dtree_tab.

Accuracy.

eg:- From sklearn.metrics import accuracy_score
accuracy_score(cr_test_y, dec_pred)

Feature Selection

Feature Importance

eg:- dtree.feature_importances_
eg:- dtree.feature_importances_.sum()

Higher the value of feature importance more important or significant that variable or feature is

eg:- cr_train_x.columns

eg:- Feature_imp = pd.DataFrame({'Feature': cr_train_x.columns, 'Importance': dtree.feature_importances_})
eg:- Feature_imp.

eg:- Feature_imp.sort_values('Importance', ascending=False)

eg:- cr_train, cr_test = train_test_split(cr, test_size=.2)

eg:- cr_train_x = cr_train.iloc[:, [9,5,7,6,10,2,8]]
cr_train_y = cr_train.iloc[:, -1]

Date:
Page No.:

cr_test_x :
cr_test_y :

cr_test_x.head()

Decision Tree using Entropy.

eg:- From sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_split=40, class_weight='balanced')

eg:- dtree.Fit(cr_train_x, cr_train_y)

eg:- dec_pred = dtree.predict(cr_test_x)
dec_pred

eg:- confusion_matrix(cr_test_y, dec_pred)

eg:- accuracy_score(cr_test_y, dec_pred)

eg:- From six import StringIO
From IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
import pydot.

eg:- dot_data = StringIO()

Date:
Page No.:

eg:- export_graphviz(dtree, outfile = dot_data, FitHeal=True
, roundcol= True, special_characters = True, Feature_names
= cx_train_x.columns)
graph = pydotplus.graph_from_dot_data(dot_data.
getvalue())

(graph,) = pydot.graph_from_dot_data(dot_data
.getvalue())

Image(graph.create_png())

Date:
Page No.:

Day 33 Random Forest.

- RF target variable can be Binary or multiclass

- RF is collection of Decision Trees.

* How random Forest work

→ In RF the algorithm uses the three techniq
1. Ensemble
2. Bootstrapping
3. Bagging

1) Ensemble

Its nothing but N-numbers of decision tree .i.e group of N-number of samples

2) Bootstrapping

It is creating the samples with the replacement.

3) Bagging -

It is nothing but aggregate or taking the vote.

N is nothing but no of sample

Date:
Page No.:

Random Forest.

eg:- From sklearn.ensemble import RandomForestClassifier
ran_for = RandomForestClassifier()
ran_for.fit(ctg_train_x, ctg_train_y)

eg:- pred_ran = ran_for.predict(ctg_test_x)
pred_ran

eg:- tab_ran = confusion_matrix(ctg_test_y, pred_ran)
tab_ran

eg:- accuracy_score(ctg_test_y, pred_ran)

Day 34 Random Forest on GridCV (before model building)

eg:- From sklearn.tree import DecisionTreeClassifier
dt_cr = DecisionTreeClassifier()

eg:- From sklearn.model_selection import GridSearchCV

eg:- search_dict = {'Criterion': ['gini', 'entropy'],
'max_depth': (5,6,7,8,9,10)
'min_samples_split': (40,50,60,75,90,120)}

eg:- grid = GridSearchCV(dt_cr, param_grid = search_dict)
grid.fit(cr_train_x, cr_train_y)

eg:- grid.best_params_
eg:- dt_cr = DecisionTreeClassifier(criteria = max depth min samples

Date:
Page No.:

Day 35 KNN Algorithm.

* KNN (K-nearcast Neighbour):-

- KNN is supervised Learning algorithm
- Distance is used to find the neighbours.
- If distance is less then chances of being neighbour is very high.

- In KNN if you dicicding the neighbours then you are not deciding it by passing the distance but you are deciding by passing the value of K

- K in KNN is nothing but the no. of neighbour

- Euclidian Distance is used to find the distance between two records

	Income	Saving	Return	Networth
A	10000	5000	2000	100000
B	10200	4500	3100	101500

$$AB = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$= \sqrt{(10000-10200)^2 + (5000-4500)^2 + (2000-3100)^2 + (100000-101500)^2}$$

- K is the hyperparameter. (integer & positive) — k value should be

Date:
Page No.:

* How KNN actually works?

Lets say i have 500 records

	X_1	X_2	X_3	Y ---> 0 & 1
1	10	8	20	1
2				
3				
⋮				
500				

As of now i will consider the value of K=5
Now comes our test data A,

A	11	9	30	?

distance betn Record 1st & person A
$= \text{sqrt}\ (10-11)^2 + (8-9)^2 + (20-30)^2$
$= 102$

Q - The value of k will not be very less.
→ If its less then your decision would be bad

Date
Page No.:

Day 36 KNN model on Adult kNN data.

Q How to find the exact value of k in KNN?
- We will use the trade off technique.
- We will do the trade off betn accuracy and stability.
- Elbow.
- Normaly we should avoid KNN.

```
eg:- import pandas as pd
     import numpy as np
     pd.set_option('display.max_columns', None)
     pd.set_option('display.max_row', None)

eg:- A = pd.read_csv(r'adultkNN.csv')
eg:- A.head()
eg:- A.isnull().sum()
eg:- A.replace('?', pd.NA, inplace = True)
eg:- A.isnull().sum()
eg:- A.workclass.value_counts()
eg:- A.workclass = A.workclass.fillna('Private')
eg:- A.occupation.value_counts()
eg:- A.occupation = A.occupation.fillna('Prof-specialty')
eg:- A['Native-country'].value_counts()
eg:- A['Native-country = A['Native-country'].fillna('United Stat
eg:- A.isnull().sum()
eg:- A.info.
## Cleaning. (ELT)

eg:- From sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
```

penkraft

Date:
Page No.:

eg:- A[A.select_dtypes(include=['object']).columns] =
[———————].Apply(le.Fit_transform)

eg:- A.info()
eg:- A.head()

Sampling

eg:- From sklearn.model_selection import train_test_split
A_train, A_test = train_test_split(A, test_size=.2)

eg:- A_train_x = A_train.iloc[:, 0:-1]
A_train_y = A_train.iloc[:, -1]

A_test_x = ———————
A_test_y = ———————

K-Nearest Neighbors

eg:- From sklearn.neighbors import kNeighborsClassifier

eg:- Aknn = kNeighborsClassifier(n_neighbors=20, weight="distance")

eg:- Aknn.Fit(A_train_x, A_train_y)

eg:- pred_knn = Aknn.predict(A_test_x)
pred_knn

Date:
Page No.:

Confusion Matrix & Accuracy.

eg:- From sklearn.metrics import confusion_metrix
tab_knn = confusion_matrix (A_test_y, pred_knn)
tab_knn.

eg:- From sklearn.metrics import accuracy_score
accuracy_score (A_test_y, pred_knn) * 100

How to Find the value of k
For this we will plot a graph betn Acc & stabel

imp

eg:- acc = []
For k in range(1,51):
    Aknn = kNeighborsClassifier(n_neighbors = k)
    Aknn.fit(A_train_x, A_train_y)
    pred_knn = Aknn.predict(A_test_x)
    tab_knn = confusion_matrix(A_test_y, pred_knn)
    acc.append(tab_knn.diagonal().sum()*100/tab_knn.sum

eg:- k_values = list(range(1,51))

eg:- import matplotlib.pyplot as plt.

eg:- plt.plot(k_values, acc).

Date:
Page No.:

Day 37 Naive Bayes Classifier.

- This algorithm works on the conditional probability.
- Bayesian Theorem. on basis of this theorem NBC works.
- We mainly use NBC. It doesn't have hyperparameters.

Q What is Conditional Probability?

→ P(A/B) → Prob of occurance of event 'A' when event 'B' has already occured.

$$P(A/B) = \frac{P(A \cap B)}{P(B)}$$ ∴ Conditional Probability formula.

eg:- import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
pd.set_option('display.max_row', None)

eg:- s = pd.read_csv(r'svmFile.csv')
eg:- s.head()
eg:- s.shape
eg:- s.isnull().sum()
eg:- s.info()

eg:- From sklearn.preprocessing import LabelEncoder
le. = LabelEncoder()

eg:- s.matrital.value_counts()

Date:
Page No.:

eg:- s.job = le.fit_transform(s.job)
s.marital = le.fit_transform(s.marital)
s['cons.conf.idx'] = le.fit_transform(s['cons.conf.idx'])
s['emp.var.rate'] = le.fit_transform(s['emp.var.rate'])

eg:- s.education.replace({'illitrate': 0, 'unknown': 0, 'basic.6y': 1, 'basic.4y': 2, 'basic.9y': 3 ....}, inplace = True).

eg:- s.default.replace({...}, —)
s.housing.replace({...}, —)
s.loan.replace({...}, —)

eg:- s.contact = le.fit_transform(s.contact)

eg:- s.month.replace({...}, —)
s.poutcome.replace({...}, —)
s.day_of_week.replace({...}, —)
s.y.replace({...}, —)

eg:- s.info()

Sampling.

from sklearn.model_selection import train_test_split
s_train, s_test = train_test_split(s, test_size = .2)
# Over Sampling
df1 = s_train[s_train.y == 1]
df1.shape
s_train = pd.concat([s_train, df1, df1, df1])

Date:
Page No.:

eg:- s_train_x = s_train.iloc[:,0:-1]
s_train_y = s_train.iloc[:,-1]

s_test_x = ——
s_test_y = ——

Naive Bayes Classifier.

eg:- from sklearn.naive_bayes import MultinomialNB
ct_s = MultinomialNB()

eg:- ct_s.fit(s_train_x, s_train_y)

eg:- pred_ct = ct_s.predict(s_test_x)
pred_ct

eg:- from sklearn.metrics import confusion_matrix
confusion_matrix(s_test_y, pred_ct)

eg:- From sklearn.metrics import accuracy_score
accuracy_score(s_test_y, pred_ct)

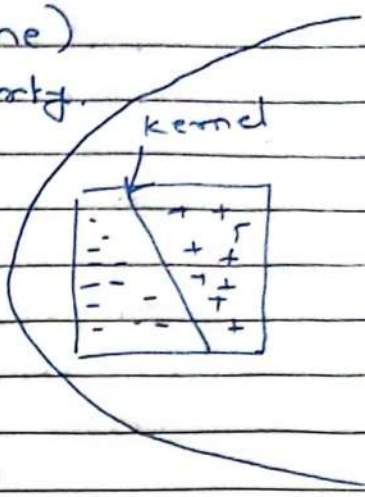
... Continue to SVM

Date:
Page No.:

Day 38 SVM Algorithm.

- SVM (Support Vector Machine)

* Supervised learning main two property.
  - Sampling is possible
  - Target variable are present.



*

eg:- from sklearn.svm import SVC
svc_s = SVC()
eg:- svc_s.fit(s_train_x, s_train_y)
eg:- pred_svc = svc_s.predict(s_test_x)
pred_svc
eg:- confusion_matrix(s_test_y, pred_svc)
eg:- accuracy_score(s_test_y, pred_svc)

Date:
Page No.:

Day 39 Solving the ML paper-1

eg:- import pandas as pd
import numpy as np

From sklearn.preprocessing import LabelEncoder

From sklearn.ensemble import RandomForestClassifier
From sklearn.feature_selection import RFE
import seaborn as sns

From sklearn.metrics import roc_curve
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt.
From sklearn.metrics import confusion_matrix
precision_score
recall_score
F1_score
accuracy_score
roc_auc_score

From sklearn.model_selection import train_test_split

From sklearn.linear_model import LogisticRegression
From sklearn.tree import DecisionTreeClassifier
From sklearn.ensemble import RandomForestClassifier
From sklearn.neighbors import KNeighborsClassifier
From sklearn.svm import SVC
import xgboost as XGBClassifier
From sklearn.ensemble import AdaBoostClassifier
From sklearn.naive_bayes import GaussianNB
From sklearn.naive_bayes import MultinomialNB

Date:
Page No.:

```
import warnings
warnings.filterwarnings('ignore')
```

eg:-
```
data = pd.read_csv(r"bank.csv", sep = ';')
data.shape
data.head(10)
data.info()
```

Q1) What does the primary analysis of several categorical feature reveal.

Day 40 Solving ML paper part-2

Day 41 Discuss Question Paper

Date:
Page No.:

Day 42 Unsupervised Learning

1. kmeans
2. PCA
3. Hierarchical Clustering
4. VIF.

— kmeans

Properties of unsupervised :-
1. Target variable is NOT present
2. Sampling is not possible.

Here in kmeans k is the no. of clusters (Groups)
This groups are formed based upon similarity.

Q How to measure the similarity.?
Distance will help us to Find similarity.
↓
Euclidean Distance.

Important -
1. kmeans unsupervised algorithm where k stands for no. of clusters.
2. k in kmeans is a hyperparameter
3. kmeans is an iterative process (Repeated)
→ kmeans will randomly select any 2 records and mark them as centroid.

25 63 37 45 52 41 29 33 31 56 44 42 53 59
27 34 38 50 28

Date:
Page No.:

- At the starting centroids were assigned Randomly.
- According to centroid we will Find the distance between centroind and the record whichever record has the least distance with centroid it will get added in that cluster.
- On 2nd iteration centroids were calculated by taking the "MEAN"

Stopping Criteria. -

When mean gets kind of Fixed which means record will stop to move From the other cluster which means our model get converged The model will stopped the

- --- In Final model will always be similar For all.

penkraft

Date:
Page No.:

Day 43 Kmeans on SNS Data.

eg:- import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
pd.set_option('display.max_row', None)

eg:- sns = pd.read_csv(r'sns.csv')
eg:- sns.head()
eg:- sns.shape

Problem Statement
The many millions of teenage cosumers using social media or networking sites have attracted the attention of marketers
So this companies or clients can avoid targetting ads to teens coith no interest in the products being sold
Build kmeans clustering model to classify the interest of teenagers by using variouy attributes

eg:- sns.info()
eg:- sns.isnull().sum()

eg:- sns.gender.value_counts()
sns.gender = sns.gender.Fillna('F')

eg:- sns.age.value_counts()
sns.age = sns.age.Fillna(sns.age.mean())

eg:- sns.isnull().sum()

Date:
Page No.:

eg:- sns.gender.replace({'F':1,'M':0} inplace=True)

sns.info()

Unsupervised model does not required sampling

eg:- From sklearn.cluster import KMeans
eg:- kmeans_sns = KMeans(n_cluster=4)
eg:- kmeans_sns.Fit(sns)

If 4 clusters have been created so each record should have gone into one of the cluster

eg:- kmeans_sns.labels_
eg:- len(kmeans_sns.labels_)

For each record u have 1 label bcoz its one on one mapping

eg:- k means_sns.cluster_centers_
Here we got 40 centroid point bcoz our data has 40 column i.e. it is 40 dimensions
So each column has seperate coordinate so we get 40 values.

eg:- sns_new = pd.DataFrame(kmeans_sns.cluster_cent

for Finding the best value or optimum value of k we need to plot the elbow plot

SSD (Sum of Squared dimension)

Date:
Page No.:

We will build the plot
run a loop from k=1 to k=10

eg:- kmeans_sns.score(sns)
While building the plot we need to ignore the minus sign i.e. we are going to take the mod.

eg:- sn = range(1, 11); print(sn)
kmeans_sns = [KMeans(n_clusters = i) for i in sn]
kmeans_sns
score = [kmeans_sns[i].fit(sns).score(sns) * -1 for i in range(len(kmeans_sns))]
print(score)

eg:- import matplotlib.pyplot as plt.

eg:- plt.plot(sn, score, marker = '*')
plt.grid()

eg:- len(sns_new)

eg:- sns['Label'] = kmeans_sns.labels_

eg:- sns.head(15)

Date:
Page No.:

Day 44 PCA Algorithm.

- PCA (Principal Component Analysis.)
- PCA is Unsupervised Machine Learning Algorithm
- PCA is the Dimensionality reduction algorith
- Dimensionality means large data, etc.
- PCA will always works on numeric data.
- PCA uses orthogonal transformation For dimensionality reduction

* Let say if 'A' is matrix and 'At' its transpose,
  A * At = Identity matrix.

here, $A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ $At = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$

but normally $A * A^{-1}$ = Identity matrix and
$A * At$ = Identity matrix
So basically transpose (At) is also acting as inverse (A^{-1})

OT. $A * X = lamda * X$
$N*N * N*1 = lamda * N * 1$

- lamda is nothing but here eigen value
  X is eigen vector (N*1)
  A = (N*N) matrix.

- PCA can also be used when you have the problem of multicolinearity.

Date:
Page No.:

- PCA can be used for dimensionality reduction but PCA cannot be used for feature selection

- In PCA the first principal component is always very important followed by the 2nd one and 3rd one

Date:
Page No.:

Day 45 PCA on PPT Data.

```
eg:- import pandas as pd
     import numpy as np
     pd.set_option('display.max_columns', None)
     pd.set_option('display.max_row', None)

eg:- ppt = pd.read_csv(r"Property_Price_Train.csv")
eg:- ppt.head()

# Data Cleaning
eg:- ppt.shape
eg:- ppt.info()
eg:- ppt.isnull().sum()
eg:- len(ppt.isnull().sum()[ppt.isnull().sum()>0])
eg:- ppt.isnull().sum()[ppt.isnull().sum()*100/ppt.shape[0]>40

eg:- ppt.Lot_Extent.value_counts()
eg:- ppt.Lot_Extent = ppt.Lot_Extent.Fillna(ppt.Lot_Extent.me

eg:- ppt.Brick_Vector_Type = ppt.Brick_Vector_Type.Fillna('Non
     ppt.Brick_Vector_Area = ppt.Brick_Veneer_Area.Fillna(ppt.
     Brick_Veneer_Area.mean())

eg:- ppt = ppt.drop(['Pool_Quality','Fence_Quality','Miscellaneou
     _Feature','Fireplace_Quality','Lane_Type'], axis=1)

eg:- ppt.shape
eg:- len(ppt.isnull().sum()[ppt.isnull().sum()>0])
eg:- ppt = ppt.iloc[:,1::]
```

penkraft

Date:
Page No.:

eg:- ppt.head()

Label Encoder

eg:- from sklearn.preprocessing import LabelEncoder
eg:- le = LabelEncoder()
eg:- ppt[ppt.select_dtypes(include=['object']).columns] = ppt[ppt.select_dtypes(include=['object']).columns].apply(le.fit_transform)
eg:- ppt.head()
eg:- ppt.info()

=

eg:- ppt1 = ppt
eg:- ppt = ppt.drop('Sale_Price', axis=1)
eg:- ppt.shape
eg:- ppt = ppt.sample(frac=1, axis=1)

↑

This code is to shuffle the columns to change the position of the columns.

PCA

eg:- from sklearn import decomposition
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

Before doing or applying the PCA, we will make our data unitless

Date:
Page No.:

eg:- scaler = StandardScaler()
eg:- scaled_ppt = scaler.fit_transform(ppt)
eg:- scaled_ppt
eg:- scaled_ppt.shape
eg:- pca_ppt = PCA()

eg:- data_after_pca_transformation = pca_ppt.fit_transform(scaled_ppt)
data_after_pca_transformation

eg:- pca_ppt.explained_variance_ratio_
eg:- list(pca_ppt.explained_variance_ratio_)
eg:- pca_ppt.explained_variance_ratio_.sum()
eg:- len(pca_ppt.explained_variance_ratio_)

eg:- l1 = list(pca_ppt.explained_variance_ratio_)
eg:- np.sum(l1[0:53])

In any model there always be an error (5 to 10%)
We can select those number of PCA which are able to explain around 90% of the variance
We can ignore the rest of them.

PCA can be use for dimentionality reduction not for Feature selection

We will build our model using our first 55 PCA's

eg:- type(data_after_pca_transformation)

penkraft

Date:
Page No.:

eg:- df1 = pd.DataFrame(data_after_pca, transformation)
df1.head()

eg:- From sklearn.linear_model import LinearRegression
eg:- lr = LinearRegression()
eg:- lr.fit(df1.iloc[:,0:53], ppt1.Sale_Price)
eg:- lr.score(df1.iloc[:,0:53], ppt1.Sale_Price)

PCA can be use to solve the problem of Multicolinearity.

eg:- df1.corr()

Date:
Page No.:

Day 46 Hirarchical Clustering

- As we increase the distance, record which are far away will start to come in cluster
- As we decrease the distance then no of cluster will increase & records in cluster will decrease
- In hierarchical clustering everything thing depends on the distance.
- We use linkage function

```
eg:- import pandas as pd
     import numpy as np
     pd.set_option('display.max_columns',None)
     pd.set_option('display.max_rows',None)

eg:- mk = pd.read_csv(r'mall_kmeans.csv')
eg:- mk
eg:- mk.shape
eg:- mk.isnull().sum()
eg:- del mk['CustomerID']
eg:- mk.info()
eg:- mk.Genre.replace({'Female':0,'Male':1},inplace=True
eg:- mk.head()

eg:- import scipy

eg:- from scipy.cluster.hirarchy import dendrogram,linkag
```

Date:
Page No.:

eg:- z1 = linkage(mk, 'ward')
eg:- z1.shape
eg:- import matplotlib.pyplot as plt

eg:- plt.figure(figsize=(10,10))
dendrogram(z1, p=16, truncate_mode='lastp')
plt.title('Hirarchical Clustering')
plt.xlabel('Cluster size')
plt.ylabel('distance')

Agglomerative Clustering

eg:- from sklearn.cluster import AgglomerativeClustering
clusters = AgglomerativeClustering(n_clusters=5)
↑# Here default value is 2

eg:- clusters.fit_predict(mk)

Lets plot them and see how clusters are distributed.

eg:- plt.scatter(mk['Age'], mk['Spending Score (1-100)'], c=clusters.labels_, cmap='rainbow')

eg:- fig, ax = plt.subplots()
sc = ax.scatter(mk['Annual Income (k$)'], mk['Spending Score (1-100)'], c=clusters.labels_, cmap='rainbow')
ax.legend(*sc.legend_elements(), title='clusters')

Date:
Page No.:

Day 47 Time Series Part 1.

- Time series is a basically a Univariate Time series. Univariate means only one variable is present.

- When we compared time series to other ML algorithms like LR, Logistic, DT, RF & KNN we were having the seperate X & Y variables

- But in time series there is no seperate X & Y variable. A single series will have both X & Y variable.

- For forecasting we use time series (eg:- Weather forcasting, stock market, etc)

For eg:- We want to do the monthly forecast of the sales, here I will be having the data From Jan to June,

x variable → Jan Feb Mar Apr May Jun

y variable → July

Jan	Feb	Mar	Apr	May	Jun	July
1000	1100	9500	8700	12000	14000	???

Q Why we need Forcasting ?

1 Patterns in Time Series -

i) Trend
ii) Seasonality
iii) Cyclic
iv) Random.

penkraft

Date:
Page No.:

1) Trend –

When the values will either increase or decrease with respect to time.

A) Uptrend :- If values are continuously increasing then we called it as uptrend.

B) Down trend :- If value are —"— decreasing as Down

2) Seasonality :-

The values will rise & Fall with Fixed time period.

3) Cyclic :-

The value will increase or decrease but not with Fixed time period and generate time period is more than 2 years.

4) Random :-

If time series is not in trend nor seasonal and not even cyclic then we called random.

eg Stock Market.

* When values are random no time series model should be created.

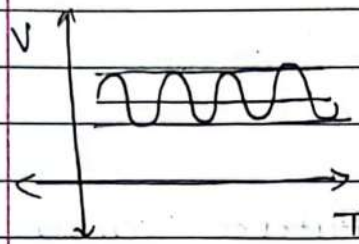
imp

Date:
Page No.:

Day 48 Stationarity in Time Series.

* Stationarity in Time Series :-

- Stationarity is the main assumption in Time Series.
- mean or the varient are time varient they do not change with the time



Variance constant / Not changing
mean constant / Not changing

- Before building the model
- If time series is not stationary then we will make it stationary.
- To make the series stationary we are going to use the Differencing Technique.

* Time series models :-
- 6 time series model

1) Auto regressive
2) Moving average
3) ARMA
4) ARIMA
5) SARIMA
6) SARIMAX.

Date:
Page No.:

Day 49 Time Series model Part 1.

1) Auto Regressive (AR) :-

In AR model Future values are forecasted using the past values.

Eqn,

$$y_t = C + \gamma_1 y_{t-1} + \gamma_2 y_{t-2} + \cdots + \gamma_n y_{t-n}$$

(y_{t-1}: Previous Time period)

- we consider the latest value

2) Moving Average :-

In MA model Future values are forecasted using the past error terms.

- Compare Actual & Predicted values & get error.

Eqn,

$$y_t = \gamma_1 E_{t-1} + \gamma_2 E_{t-2} + \cdots + \gamma_n E_{t-n}.$$

3) ARMA :- (AR + MA.)

- We are going to do the forecast using the previous actual values + previous error term also. ARMA (p, q)

- Here AR can be mapped to 'p' & MA can be mapped to 'q'

Date:
Page No.

Day 50 ARIMA and Differenced Technique.

4) ARIMA Model
(Auto Regressive integrated moving average)

ARIMA can be denoted as (p(AR), d(I), q(MA))

* Lets try to understand what is integreted mean?
  - If series is not stationary we will make it stationary
  - Differencing Technique

Let say i have a series;
200, 210, 217, 240, 280, 360, 400, 205, 213, 220, 245, 300, 370. and this is not stationary.

We will be using Differencing Technique to make it stationary

Differencing means substracting 1 term for the other terms.
After differencing,
10, 7, 23, 40, 80, 40, -195, 8, 7, 25, 55, 70.
... series after differencing

You will do 2nd level of differencing if series is not sationary even after 1st level of differencing
-3, 16, 17, 40, -40, -155 d
d= it will be decided by the level of differencing you are doing

Date:
Page No.:

- How to Find the value of p & q

By using ACF & PACF
1) ACF (Auto correlation Function)
2) PACF (Partial Auto Correlation Function)

g:- jan feb mar April may
540 510 530 540 580

The correcalation will come from the lag of the series.

Time Series on Air Passengers.

Date:
Page No.:

Day 51, 52, 53.

eg:- import pandas as pd
import numpy as np.

eg:- airpas = pd.read_csv(r"AirPassengers.csv")
eg:- airpas

PS is to Forecast the passengers From next 12 months (For year 1961)
Random sampling is crime
Breaking the series is also crime.

eg:- airpas.info()

eg:- airpas.Month = pd.to_datetime(airpas.Month, infer_datetime_format = True)

eg:- airpas.info()

eg:- airpas = airpas.set_index(['Month'])
airpas

~~eg:-~~

We will check if our series is stationary or not.

eg:- import matplotlib.pyplot as plt.

eg:- plt.plot(airpas)

Date:
Page No.:

the series which we got is not stationary and we have to make it stationary

eg:- plt.plot(airpas.diff().diff().diff())

Even after multiple level of differencing series does not became stationary
We are going to take the log of the series.

eg:- airpas_log = np.log(airpas)
g:- airpas_log

eg:- plt.plot(air_log)
eg:- plt.plot(air_log.diff().diff().diff())

After taking log and applying differencing we get the series stationary
If you are using the log your model variance & mean will be constant.
for building a model we need 3 thing ars p, d, q and we know (p=?, d=1, q=?

To find the p & q we need to use ACF & PACF.

eg:- from statsmodels.graphics.tsplots import plot_acf, plot_pacf.

eg:- plot_acf(airpas_log)

eg:- plot_pacf(airpas_log)

penkraft

Date:
Page No.:

When your ACF declines slowly or gradually and PACF declines sharply then we called it as AR process
Here are 2 lags we are getting correlation =0 so that is AR(2) process
AR is Mapped to p
The value we got = (p=2, d=1, q=0)

ARIMA.

eg:- from statsmodels.tsa.arima_model import ARIMA.

The problem which associated with ARIMA
ARIMA can not be handled the seasonality

SARIMA
Seasonal Auto Regressive Integrated moving average.

SARIMAX.
Seasonal Auto Regressive Integrated moving average with Exogenous Variables

How to use SARIMA & SARIMAX

We will use auto-arima and whatever values we will get from auto arima we are going to pass it into our model.

eg:- from statsmodels.tsa.statespace.sarimax import SARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
import pmdarima
from pmdarima import auto_arima

Date:
Page No.:

eg:- result = seasonal_decompose (airpas_log)
eg:- result.plot();
# the plot is to just get the idea about data

eg:- auto_arima (airpas_log, seasonal = True, m=12)
m is the time period

Whatever values will get from the auto_arima we will pass them in SARIMAX.

eg:- model_sarima = SARIMAX (airpas_log, order = (2,0,0), seasonal_order = (0,1,1,12))

eg:- airpas_model = model_sarima.fit()

eg:- pred_values = airpas_model.predict (start = 144, end=[illegible]
pred values

eg:- prd_values1 = np.exp(pred_values)
prd_values1

eg:- plt.plot (airpas_log)
plt.plot (pred_values)

eg:- df = pd.DataFrame()
df['Predicted'] = pred_values1
date_for_pred = ['1961-01-01 ..... '1961-02-01']
df['Month'] = date_for_pred
df.Month = pd.to_datetime (df.Month, infer_datetime_format = True)
df = df.set_index (['Month'])

penkraft

Date:
Page No.:

g:- df.

g:- airpay['Passengers'].plot(legend=True, label='Orig
,color='green')
df['Predicted'].plot(legend=True, label='Predicted'
color='red')

Day 54 Time series on Alcohol Sales