# HOUSE PRICE PREDICTION PROJECT

## Q1.what all libraries were used in your project,describe each?

1. *NUMPY ( DATA PROCESSING AND MODELLING)*

NumPy (Numerical Python) is a perfect tool for scientific computing and performing basic and advanced array operations. The library offers many handy features **performing operations on n-arrays and matrices** in Python. It helps to process arrays that store values of the same data type and makes performing math operations on arrays (and their vectorization) easier. In fact, the **vectorization of mathematical operations on the NumPy array type increases performance** and **accelerates the execution time.**

1. *MATPLOTLIB (DATA VISUALIZATION)*

This is a standard data science library that helps to generate data visualizations such as

two-dimensional diagrams and graphs **(histograms, scatterplots, non-Cartesian coordinates graphs)**. Matplotlib is one of those plotting libraries that are really useful in data science projects — it provides an object-oriented API for embedding plots into applications.

It's thanks to this library that Python can compete with scientific tools like MatLab or Mathematica. However, developers need to write more code than usual while using this library for generating advanced visualizations(HENCE SEABORN IS USED). Note that popular plotting libraries work seamlessly with Matplotlib.

1. *SEABORN(DATA VISUALISATION)*

Seaborn is based on Matplotlib and serves as a useful Python machine learning tool for **visualizing statistical models – heatmaps and other types of visualizations that summarize data and depict the overall distributions.** When using this library, you get to

benefit from an **extensive gallery of visualizations** (including complex ones like time series, joint plots, and violin diagrams).

1. *PANDAS(DATA PROCESSING AND MODELLING)*

Pandas is a library created to help developers work with "**labeled" and "relational" data** intuitively. It's based on two main data structures: **"Series" (one-dimensional, like a list of items) and "Data Frames" (two-dimensional, like a table with multiple columns)**. Pandas allows **converting data structures to DataFrame objects, han**

**dling missing data, and adding/deleting columns from DataFrame, imputing missing files**, and plotting data with histogram or plot box. It’s a must-have for data wrangling, manipulation, and visualization.

**5)SciPy**

The SciPy library is one of the core packages that make up the SciPy stack. It provides many user-friendly and efficient numerical routines, such as routines for numerical integration, interpolation, optimization, linear algebra, and statistics.

#import libraries

import numpy as np

import matplotlib.pyplot as plt import pandas as pd

import seaborn as sns import statistics as st

from scipy.stats import norm

from sklearn.preprocessing import StandardScaler from scipy import stats

from scipy.stats import norm, skew

## Q2. what was the data about ,how many features were there and describe your data?

Over 80 independent features were used in the data set to predict the sales price of 1500 houses approximately. It's real life data taken from the Kaggle website. There were many missing values[if numerical data, we filled it using mean][if categorical, we filled it using mode] , categorical data features[labelencoder(disadv), one hot encoding ,dummy variable trap] and outliers (by plotting scatter plots and manually examining it ) and hence thorough cleaning was done. Using backward elimination, features that did not impact the house price much were removed. Using correlation matrix, correlation of every feature with every other feature was examined and the ones which correlated more than 90% were removed to avoid **multicollinearity [situation where 2 or more IVs are highly related. This leads to lack of statistical significance to individual IVs although the overall model is significant.] Hence, avoiding multicollinearity is crucial.**

Why target variable is should be normally distributed?

In some cases, it may actually help getting better results (depending on the model type), but it is also likely that the improvement comes from the fact that the performance metric is computed differently. For instance, a skewed distribution will lead to high MSE values due to cases located on the other side of the distribution, while the MSE is limited if the data is transformed to a normal distribution. So when comparing the cases, make sure you evaluate the performance on the back-transformed target.

Cases where the model will actually perform better with a normally distributed target include, among others, Gaussian process regression, because of the underlying assumption of a Gaussian random variable. There should be quite a few other model types which somehow have similar assumptions, and thus perform better with transformed data.

#importing dataset

trainset= pd.read\_csv('train.csv') testset=pd.read\_csv('test.csv')

"""DATA CLEANING AND PREPROCESSING"""

**#Identifying and dropping the outliers** x\_gr=trainset.loc[:,'GrLivArea'].values y\_sp=trainset.loc[:,'SalePrice'].values plt.scatter(x\_gr, y\_sp, color='red') plt.title('GalLivArea vs Salesprice') plt.show()

#drop outliers

trainset = trainset.drop(trainset[(trainset['GrLivArea']>4000) & (trainset['SalePrice']<300000)].index) x\_bsmt=trainset.loc[:,'TotalBsmtSF'].values y\_sp=trainset.loc[:,'SalePrice'].values

plt.scatter(x\_bsmt, y\_sp, color='red') plt.title('TotalBsmtSF vs Salesprice') plt.show()

#drop outliers(There were no more outliers)

**Q3. What are outliers and why should they be removed and how did you remove them ?**

A value that "lies outside" (is much smaller or larger than) most of the other values in a set of data. Basically, outliers are data points that go against the logic of the whole dataset. They should be removed. Outliers increase the variability in your data, which decreases statistical significance. If variability is more, overfitting happens.

OVERFITTING is dangerous. The quality of the results worsens when you try to learn too much from a sample.

**Overfitting** happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model.

We removed them manually.

**Q4.Why Feature Selection?**

Three key benefits of performing feature selection on your data are:

* + **Reduces Overfitting**: Less redundant data means less opportunity to make decisions based on noise.
  + **Improves Accuracy**: Less misleading data means modeling accuracy improves.
  + **Reduces Training Time**: Less data means that algorithms train faster.

**#ID feature is not useful hence drop it** trainset = trainset.drop(['Id'], axis=1) testset = testset.drop(['Id'], axis=1)

#descriptive statistics summary

trainset['SalePrice'].describe()

#histogram PLOT

sns.distplot(trainset['SalePrice'])

#The target variable is positively skewed, we need to make the data more #normally distributed so we need logarithmic trasformation """Mode(1.4)<Median(1.6)<Mean(1.8) Hence it is Postively skewed""" y\_sp=trainset.loc[:,'SalePrice'].values

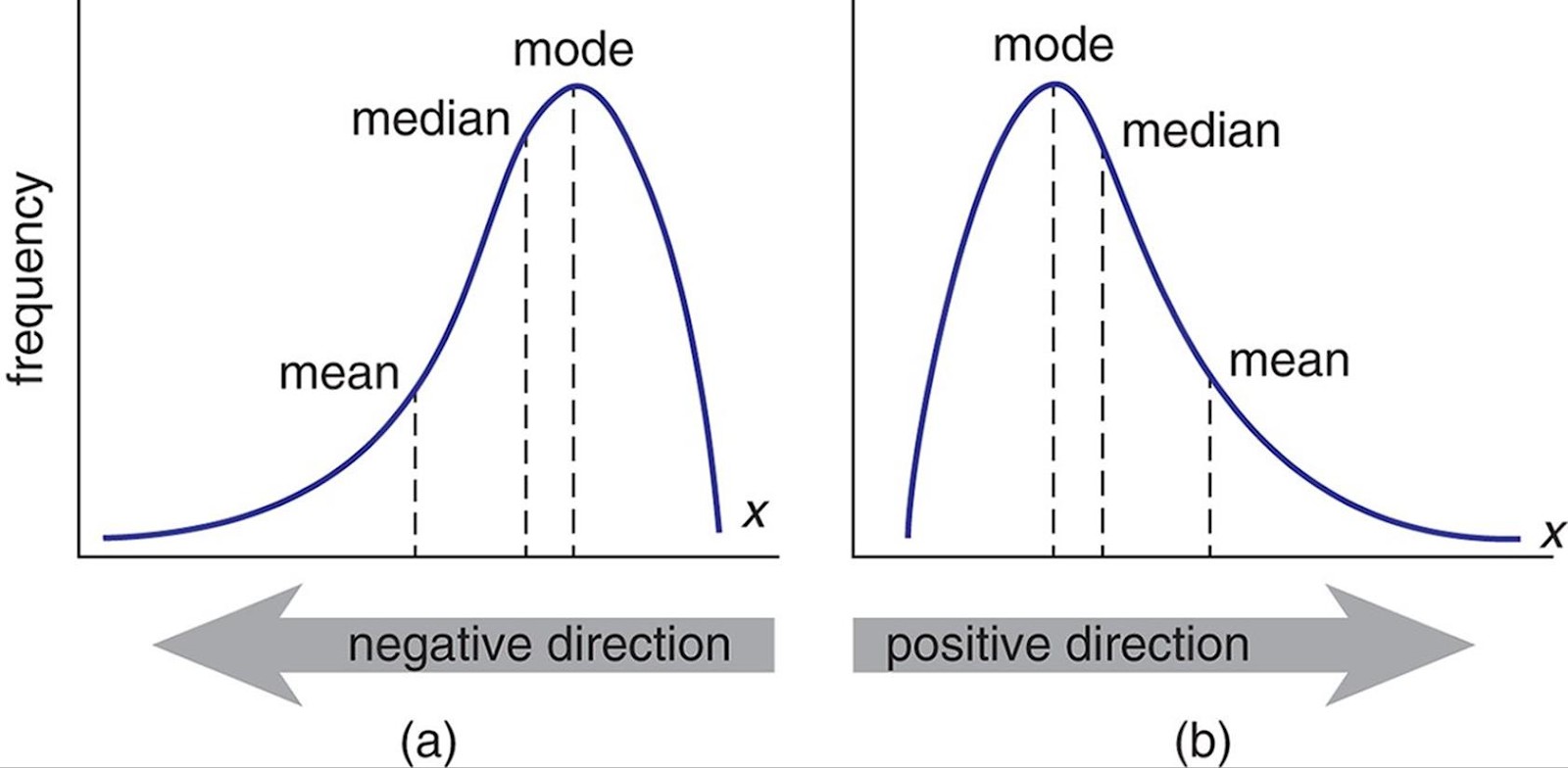
st.median(y\_sp) st.mode(y\_sp)

trainset['SalePrice']=np.log(trainset['SalePrice']) sns.distplot(trainset['SalePrice'])

## Q5.WHAT IS SKEWNESS

Skewness: it is a measure of asymmetry of data. It indicates whether data is concentrated on one side or no. Negative skew commonly indicates that the *tail* is on the left side of the distribution, and positive skew indicates that the tail is on the right.

NEGATIVE SKEW : mean < median POSITIVE SKEW : mean > median



### “””Missing Data”””

#### CONCATENATION: merging things to make them one.

#Concatenate test and train datasets

columns = pd.concat([trainset, testset]).reset\_index(drop=True)

#We need to ﬁnd out no.of nulls under each feature

nulls = np.sum(columns.isnull())

#Columns with nulls

nullcols = nulls.loc[(nulls != 0)]

#Finding out the data types of all features

dtypes = columns.dtypes

#Finding out the data types of each feature which has missing values

dtypes2 = dtypes.loc[(nulls != 0)]

info = pd.concat([nullcols, dtypes2], axis=1).sort\_values(by=0, ascending=False) print(info)

print("There are", len(nullcols), "columns with missing values")

**#Filling CATEGORICAL missing value with most frequency one** columns['Electrical'] = columns['Electrical'].ﬁllna(columns['Electrical'].mode()[0]) columns['Exterior1st'] = columns['Exterior1st'].ﬁllna(columns['Exterior1st'].mode()[0]) columns['Exterior2nd'] = columns['Exterior2nd'].ﬁllna(columns['Exterior2nd'].mode()[0])

columns['SaleType'] = columns['SaleType'].ﬁllna(columns['SaleType'].mode()[0]) columns['KitchenQual'] = columns['KitchenQual'].ﬁllna(columns['KitchenQual'].mode()[0]) columns['Functional'] = columns['Functional'].ﬁllna(columns['Functional'].mode()[0])

#To check if Pool Area>0 but Pool Quality is absent

actual\_missing\_pools = columns[columns['PoolArea'] > 0 & columns['PoolQC'].isnull()]

**#Filling the three PoolQ missing values** columns.loc[2418, 'PoolQC'] = 'Fa' columns.loc[2501, 'PoolQC'] = 'Gd' columns.loc[2597, 'PoolQC'] = 'Fa'

**#Garage missing data ﬁlling** actual\_missing\_garages=columns[columns['GarageType'].notnull() & columns['GarageYrBlt'].isnull()]

columns.loc[2124, 'GarageYrBlt'] = columns['GarageYrBlt'].median() columns.loc[2574, 'GarageYrBlt'] = columns['GarageYrBlt'].median()

columns.loc[2124, 'GarageFinish'] = columns['GarageFinish'].mode()[0] columns.loc[2574, 'GarageFinish'] = columns['GarageFinish'].mode()[0]

columns.loc[2574, 'GarageCars'] = columns['GarageCars'].median()

columns.loc[2124, 'GarageArea'] = columns['GarageArea'].median() columns.loc[2574, 'GarageArea'] = columns['GarageArea'].median()

columns.loc[2124, 'GarageQual'] = columns['GarageQual'].mode()[0] columns.loc[2574, 'GarageQual'] = columns['GarageQual'].mode()[0]

columns.loc[2124, 'GarageCond'] = columns['GarageCond'].mode()[0] columns.loc[2574, 'GarageCond'] = columns['GarageCond'].mode()[0]

#Taking care of basement related features

basement\_columns = ['BsmtHalfBath','BsmtFullBath','BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF']

tempdf = columns[basement\_columns] tempdfnulls = tempdf[tempdf.isnull().any(axis=1)]

columns['BsmtFinSF1'] = columns['BsmtFinSF1'].ﬁllna(0) columns['BsmtFinSF2'] = columns['BsmtFinSF2'].ﬁllna(0) columns['BsmtUnfSF'] = columns['BsmtUnfSF'].ﬁllna(0) columns['TotalBsmtSF'] = columns['TotalBsmtSF'].ﬁllna(0) columns['BsmtHalfBath'] = columns['BsmtHalfBath'].ﬁllna(0)

columns['BsmtFullBath'] = columns['BsmtFullBath'].ﬁllna(0) actual\_missing\_basements = tempdfnulls[(tempdfnulls.isnull()).sum(axis=1) < 5]

columns.loc[332, 'BsmtFinType2'] = 'ALQ' #smaller area than SF1 columns.loc[947, 'BsmtExposure'] = 'No' #Unﬁnished basement columns.loc[1485, 'BsmtExposure'] = 'No' #Unﬁnished basement columns.loc[2038, 'BsmtCond'] = 'Gd'

columns.loc[2183, 'BsmtCond'] = 'Gd' columns.loc[2215, 'BsmtQual'] = 'Po' #Less area columns.loc[2216, 'BsmtQual'] = 'Fa' #bit better

columns.loc[2346, 'BsmtExposure'] = 'No' #Unﬁnished basement columns.loc[2522, 'BsmtCond'] = 'Gd'

#MSSubClass and MSZoning are related

tempdf = columns[['MSSubClass','MSZoning']]

actual\_missing\_MSZoning = tempdf[tempdf.isnull().any(axis=1)] columns.loc[1913, 'MSZoning'] = 'RM'

columns.loc[2214, 'MSZoning'] = 'RL' columns.loc[2248, 'MSZoning'] = 'RM' columns.loc[2902, 'MSZoning'] = 'RL' **#Above ﬁlling is based on MSSubClass**

#Filling the Missing categorical features by None

cfeatures = [ ]

for i in columns.columns:

if columns[i].dtype == object: cfeatures.append(i)

columns.update(columns[cfeatures].ﬁllna('None'))

#LotFrontage

neighborhood = columns.groupby('Neighborhood') lot\_medians = neighborhood['LotFrontage'].median() lot\_medians

columns['LotFrontage'] = columns.groupby('Neighborhood')['LotFrontage'].transform(lambda x: x.ﬁllna(x.median()))

#MasVnrArea missing will be zero as MasVnrtype is none

columns['MasVnrArea'] = columns['MasVnrArea'].ﬁllna(0)

#GarageYrBlt missing values can be zero as no garage is there

columns['GarageYrBlt'] = columns['GarageYrBlt'].ﬁllna(0)

nulls = np.sum(columns.isnull()) nullcols = nulls.loc[(nulls != 0)] dtypes = columns.dtypes dtypes2 = dtypes.loc[(nulls != 0)]

info = pd.concat([nullcols, dtypes2], axis=1).sort\_values(by=0, ascending=False) print(info)

print("There are", len(nullcols), "columns with missing values") column\_charecteristics = columns.describe()

#there is a year 2207 in GarageYrBlt

tempdf = columns[columns['GarageYrBlt']==2207]

#This house was built in 2006 and was sold in 2007 so mostly input error for 2007

columns.loc[2590, 'GarageYrBlt'] = 2007

columns = pd.get\_dummies(columns).reset\_index(drop=True) columns.shape

num\_features = columns.dtypes[columns.dtypes != "object"].index

**# Check the skew of all numerical features** skewed\_features = columns[num\_features].apply(lambda x: skew(x.dropna())).sort\_values(ascending=False) print("\nSkew in numerical features: \n")

skewness = pd.DataFrame({'Skew' :skewed\_features}) skewness

skewness = skewness[abs(skewness) > 0.75]

print("There are {} skewed numerical features to Box Cox transform".format(skewness.shape[0]))

#from scipy.special import boxcox1p

skewed\_features = skewness.index

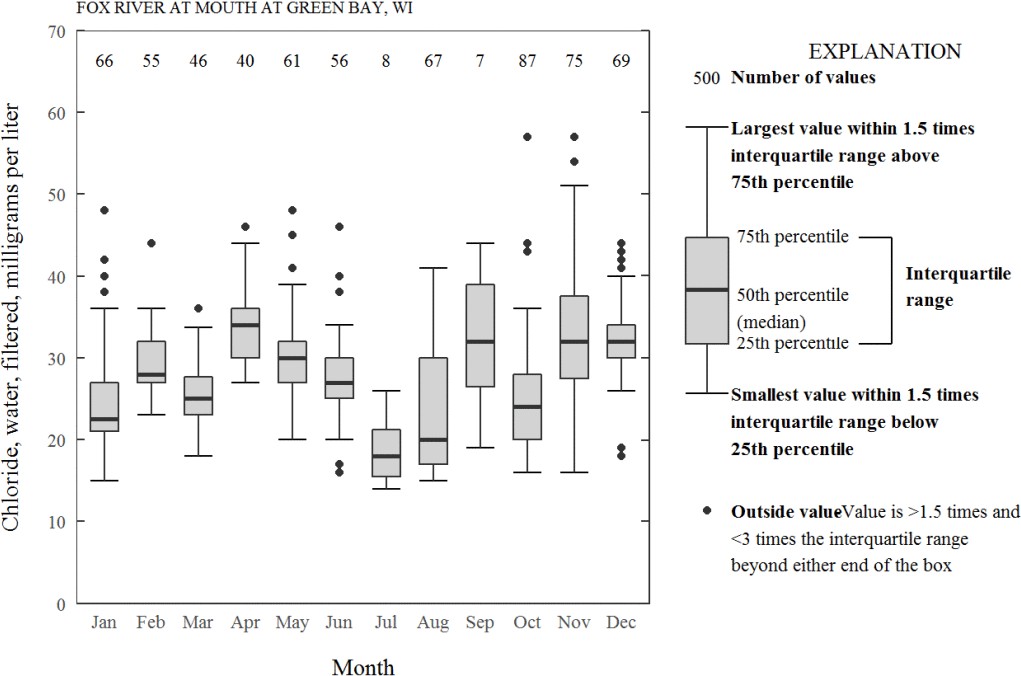
#lambda = 0.15

for features in skewed\_features: if features!="SalePrice":

columns[features] = np.log1p(columns[features])

## Q4.BOXPLOT (seaborn)

Boxplot is a method for graphically depicting groups of numerical data through their quartiles.



## Q6. HOW TO CONVERT SKEWED DISTRIBUTION TO A NORMAL DISTRIBUTION?

1. LOG TRANSFORM

Log transformation is most likely the first thing you should do to remove skewness from the predictor. It can be easily done via Numpy, just by calling the log() function on the desired column. You can then just as easily check for skew.

1. SQUARE ROOT TRANSFORM
2. BOX COX TRANSFORM

A Box Cox transformation is a way to transform non-normal dependent variables into a normal shape. Normality is an important assumption for many statistical techniques

At the core of the Box Cox transformation is an exponent, lambda (λ), which varies from -5 to 5. All values of λ are considered and the optimal value for your data is selected; The “optimal value” is the one which results in the best approximation of a normal distribution curve. The transformation of Y has the form: (FOR POSITIVE VALUES OF LAMBDA)

“””NOTE: WE COMBINED CONCATENATED TEST AND TRAINING DATA TO TAKE CARE OF FEATURES OF BOTH TEST AND TRAINING DATA , EX: LOGARTHMIC TRANSOFORMATION TO BOTH IS NEEDED”””

**Q7. WHAT IS CORRELATION MATRIX**

A **correlation matrix** is a table showing **correlation** coefficients between variables. Each cell in the table shows the **correlation** between two variables. To remove the correlated features, we can make use of the corr() method of the pandas dataframe. The corr() method returns a correlation matrix containing correlation between all the columns of the dataframe. We can then loop through the correlation matrix and see if the correlation between two columns is greater than threshold correlation, add that column to the set of correlated columns. We can remove that set of columns from the actual dataset.Feature selection plays a vital role in the performance and training of any machine learning model.

**Q8. WHY should we remove features that are highly correlated. Why is it bad to have correlated features**

It is clear that correlated features means that they bring the same information, so it is logical to remove one of them.Correlated features in general don't improve models (although it

depends on the specifics of the problem like the number of variables and the degree of correlation), but they affect specific models in different ways and to varying extents:

* 1. For linear models (e.g., linear regression or logistic regression), multicollinearity is not at all good characteristic. It can yield solutions that are wildly varying and possibly numerically unstable.
  2. Random forests can be good at detecting interactions between different features, but highly correlated features can mask these interactions.

## Q9.WHAT is multicollinearity

In statistics, multicollinearity is a phenomenon in which one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy. Independent variables should be independent. If the degree of correlation between variables is high enough, it can cause problems. It becomes difficult for the model to estimate the relationship between each independent variable and the dependent variable independently because the independent variables tend to change in unison.

corrmat = columns.corr() plt.subplots(ﬁgsize=(12,9)) sns.heatmap(corrmat, vmax=0.9, square=True)

columns = columns.drop(['MiscVal',

'SaleCondition\_Partial', 'GarageQual\_None', 'GarageCond\_None', 'GarageType\_None', 'BsmtFinType2\_None', 'BsmtFinType1\_None', 'BsmtExposure\_None', 'BsmtCond\_None', 'BsmtQual\_None', 'BsmtQual\_Po', 'Exterior2nd\_Wd Sdng', 'Exterior2nd\_MetalSd', 'Exterior2nd\_VinylSd', 'Exterior2nd\_MetalSd', 'Exterior2nd\_HdBoard', 'Exterior2nd\_CmentBd', 'Exterior2nd\_AsbShng', 'MiscFeature\_Shed',

'PoolQC\_None', 'MiscFeature\_None', 'BsmtFinType1\_None', 'BsmtFinType1\_Unf', 'BsmtFinType2\_None', 'BsmtFinType2\_Unf', 'LotShape\_Reg', 'LandSlope\_Mod', 'RoofMatl\_CompShg', 'RoofStyle\_Hip', 'MasVnrType\_None', 'ExterQual\_TA', 'ExterCond\_TA', 'CentralAir\_Y', 'KitchenQual\_TA'], axis=1)

train = columns.loc[0:1457] test = columns.loc[1458:2917]

test = test.drop(['SalePrice'],axis=1) test=test.values

y=train.loc[:,'SalePrice'].values train=train.drop(['SalePrice'],axis=1) x=train.values

## Q9.What are train\_set and test\_set?

###### The train-test procedure is appropriate when there is a sufficiently large dataset available.

* **Train Dataset**: Used to fit the machine learning model.

###### **Test Dataset**: Used to evaluate the fit machine learning model.

Nevertheless, common split percentages include:

* Train: 80%, Test: 20%
* Train: 67%, Test: 33%
* Train: 50%, Test: 50%

#Spliting into train\_set and test\_set

from sklearn.model\_selection import train\_test\_split x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y, test\_size=0.2, random\_state=0)

## Q9.What all models did you use?

LINEAR REGRESSION MODEL DECISION TREE MODEL RANDOM FOREST MODEL

## Q10.What is RMS and Adjusted RMS?

#FITTING INTO SIMPLE LINEAR REGRESSION MODEL(0.112)

from sklearn.linear\_model import LinearRegression regressor = LinearRegression()

regressor.ﬁt(x\_train, y\_train) y\_predict\_SIMLIN=regressor.predict(x\_test)

**#Visualizing the training set results in SIMPLE linear reg** plt.scatter(x\_train[:,15], y\_train, color='red') plt.plot(x\_train[:,15], regressor.predict(x\_train),color='green') plt.title('SALE PRICE(TRAINING SET)')

plt.xlabel('MSsubclass') plt.ylabel('saleprice') plt.show()

#TEST ERROR

from sklearn.metrics import mean\_squared\_error error=mean\_squared\_error(y\_test, y\_predict\_SIMLIN) error=error\*\*(0.5)

error

def meanerror(y\_test,y\_predict\_SIMLIN): y\_test=np.array(y\_test) y\_predict\_SIMLIN=np.array(y\_predict\_SIMLIN)

return np.mean(np.abs((y\_test-y\_predict\_SIMLIN)/y\_test))\*100

Error=meanerror(y\_test,y\_predict\_SIMLIN)

index=[]

for i in range(0,len(y\_predict\_SIMLIN)): if y\_predict\_SIMLIN[i]>10:

index.append(i)

y\_test=np.delete(y\_test,index, axis=0) x\_test=np.delete(x\_test,index, axis=0) y\_predict\_SIMLIN=np.delete(y\_predict\_SIMLIN,index, axis=0)

# TRAINING THE DECISION TREE REGRESSION MODEL ON THE WHOLE DATASET (0.17068)

from sklearn.tree import DecisionTreeRegressor

regressorDT= DecisionTreeRegressor(criterion="mse", splitter="best", max\_depth=None,

min\_samples\_split=50,

min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0., max\_features=None, random\_state=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0., min\_impurity\_split=None, presort='deprecated', ccp\_alpha=0.0)

regressorDT.ﬁt(x\_train, y\_train) y\_predict\_dectree=regressorDT.predict(x\_test) y\_submission\_DT=regressorDT.predict(test)

#to check error in log form #y\_predict\_dectree=np.log(y\_predict\_dectree)

error=mean\_squared\_error(y\_test, y\_predict\_dectree) error=error\*\*(0.5)

error

# TRAINING THE RANDOM FOREST REGRESSION MODEL ON THE WHOLE DATASET (0.1257)

from sklearn.ensemble import RandomForestRegressor

regressorRF= RandomForestRegressor(n\_estimators = 5000, random\_state = 0)

regressorRF.ﬁt(x\_train, y\_train) y\_pred\_ranfor=regressorRF.predict(x\_test) y\_submission\_RF=regressorRF.predict(test)

#to check error in log form #y\_pred\_ranfor=np.log(y\_pred\_ranfor)

error=mean\_squared\_error(y\_test, y\_pred\_ranfor) error=error\*\*(0.5)

Error

# TABLEAU Citi-Bike trips PROJECT

### Q1)What is the project?

Visualizing Citi Bike Trips with Tableau. This is the hands-on project that I worked on in Tableau Public using Rhyme(a cloud desktop). In this project, I learnt the basics of using Tableau, which is one of the most popular business intelligence tools out there. I imported data from Github, understood what measures and dimensions are, I used filters, used sheets, and created and exported dashboards.

Q2) DESCRIBE YOUR DATA AND DATA SOURCE?

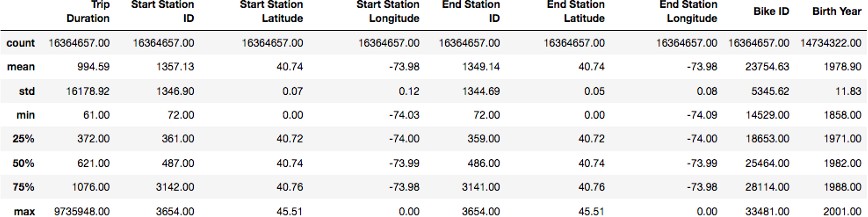
The data was sourced from the Citi Bike’s amazon server which I have taken from Github.

This dataset is massive. It has almost 14 million rows and 11 features namely:

1. Trip Duration (seconds) — How long a trip lasted
2. Start Time and Date
3. Stop Time and Date
4. Start Station Name
5. End Station Name
6. Station ID - Unique identifier for each station
7. Station Latitude/Longitude - Coordinates of station
8. Bike ID - unique identifier for each bike
9. User Type (Customer = 24-hour pass or 3-day pass user; Subscriber = Annual Member) - Customers are usually tourists, subscribers are usually NYC residents
10. Gender (Zero=unknown; 1=male; 2=female) - Usually unknown for customers since they often sign up at a kiosk
11. Year of Birth - Self entered, not validated by an ID.

**Q3)WHAT IS THE FIRST STEP?**

After acquiring data, the next step is to check if there’s any noise or cleanup which needs to be done before creating the chart. There were some missing values which were filled using python libraries using the mean method. We can’t simply drop these rows. Blindly dropping rows with missing values, we’ll be losing critical information.



**Q4) WHAT IS TABLEAU?**

**Tableau** is a data analytics platform used for Business Intelligence. It helps us to analyse, visualise and ultimately get insights from data. These insights can be used for decision making.

WORLD HAPPINESS INDEX PROJECT

**Q1) WHY DID YOU CHOOSE THIS PROJECT?**

The World Happiness Index generally indicates the level of happiness and satisfaction among the residents in a given country. The variables on which happiness of a currently generally depends include: real GDP per capita, social support, healthy life expectancy, freedom to make life choices, generosity, and perceptions of corruption. Each country is also compared against a hypothetical nation called **Dystopia**. Dystopia represents the lowest national averages for each key variable and is, along with residual error, used as a regression benchmark. As of March 2020, Finland was ranked the happiest country in the world three times in a row. Basically, Measurements of well-being can be used effectively to assess the progress of nations. It's real time data published every year by The United nations.

I choose this Because happiness is increasingly considered an important and useful way to guide public policy and measure its effectiveness. Measuring happiness is becoming an important tool. Economic growth, physical health and confidence in the future prospects of a person or country are all important factors when assessing and comparing them. **India** was at a dismal **144 rank** of a total **156** nations surveyed. So, I decided to analyse it.

**Q2) DESCRIBE YOUR DATA**

Top countries tend to have high values for all six of the key variables that have been found to support well-being: *income, healthy life expectancy, social support, freedom, trust and generosity.*

**Q3) What Makes Finland The Happiest Country In The World?**

##### Finland has extensive welfare benefits, low levels of corruption, well-functioning democracy, and its instilled sense of freedom and autonomy.

1. More than 80% of Finns trust their police force, which is far more than many other countries.
2. The country is famous for being one of the first countries to push the flat working model. The flat working model is one in which there are few - or sometimes even zero - hierarchical levels between management and staff. Typically there is less supervision of employees. It enables open communication between all departments and teams within a business. This increases workplace productivity, team-cohesion and agility. This also helps in healthy work life and personal life balance.
3. With the country’s commitment to closing the gender pay gap, as well as high-quality education, Finland is in fact the only country in the developed world in which fathers spend more time with school-aged children than mothers.
4. At a time when the gap between the rich and the poor is widening in most countries around the world, Finland has consistently worked to ensure that its poorest citizens are looked after.
5. The country’s novel ‘housing first’ principle ensures that, after being given the right support, rough sleepers can own a home of their own; a non-traditional approach to a traditional problem.

**Q4) Why are Indians unhappy despite good GDP?**

The reason is a combination of social, political, and economic stresses. Freedom of choice, life expectancy, and generosity were not highly reported values in India.

Clearly the correlation between happiness and wealth is weak. In the world, however, most happy countries are also high income countries.

1. rapid urbanisation and congestion in cities, environmental pollution and problems of commuting.
2. People worry on many fronts, especially when bringing up children in big cities. 3)There is the big question of law and order and safety regarding women.
3. There are worries about food and water safety.
4. People are insecure about their health because the question of cost of treatment plagues their minds.
5. People are happy when they have secure jobs and a regular flow of income. 7)Growing inequality. The poverty in India may have come down quite significantly but still we have millions of very poor and nearly poor who are just above the poverty line. 8)The legal system is clogged.
6. Those engaged in agriculture are unhappy about the unremunerative prices and low incomes.
7. Lack of adequate affordable housing is making millions of slum dwellers unhappy. 11)Labour forces are not happy due to job insecurity and dismal conditions of work. 12)Industrial growth is lacklustre and hence rise in unemployment.

**#Import libraries**

import numpy as np

import matplotlib.pyplot as plt import pandas as pd

import math as math import seaborn as sns

**#importing dataset**

dataset1=pd.read\_csv('FINAL.csv')

**#line chart b/w life expectancy vs countries** plt.ﬁgure(ﬁgsize=(30,5)) plt.plot(dataset1['Country'],dataset1['Health..Life.Expectancy.']) plt.title('life expectancy vs country')

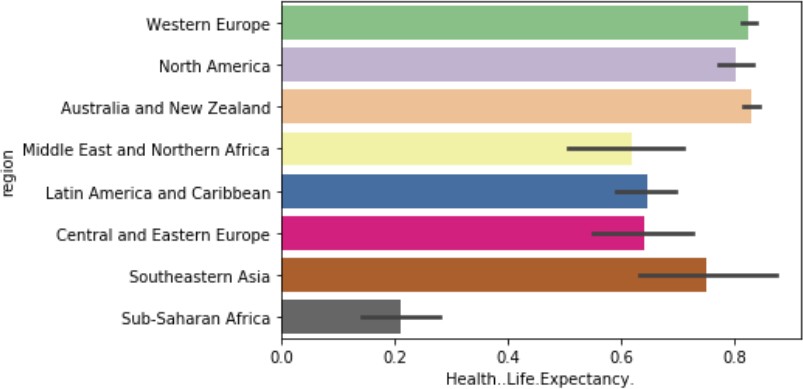
##### plt.xlabel('country') plt.ylabel('life expectancy') plt.show()

**#linchart population size vs countries** plt.ﬁgure(ﬁgsize=(30,5)) plt.plot(dataset1['Country'],dataset1['Population']) plt.title('Population vs country') plt.xlabel('country')

##### plt.ylabel('Population') plt.show()

**#Bar graph region vs life expectancy** sns.barplot(x="Health..Life.Expectancy.", y="region", data=dataset1, palette='Accent')

##### sns.barplot(x="Happiness.Score", y="region", data=dataset1, palette='Accent')



[Q1] Are there some regions that seem to live longer or less than others? [A1] FROM THE ABOVE GRAPH i.e "REGION" vs "HEALTH LIFE EXPECTANCY", we can say that there are some regions that live longer than others and some regions that live shorter than others. EXAMPLE: People of Sub-Saharan African region seem to have very less life expectancy when compared with other regions.

#### #Correlation and linear regression #Task-1:Happiness and life expectancy

##### dataset1['Happiness.Score'].corr(dataset1['Health..Life.Expectancy.'])

**# 0.9071593037581851**

**#Task-2: Population and Happiness** dataset1['Happiness.Score'].corr(dataset1['Population']) **#0.10205182030147063**

#### #Task-1:Happiness and life expectancy #Dividing into x and y

##### x = dataset1.iloc[:, 2].values y = dataset1.iloc[:, 7].values x=x.reshape(-1,1) y=y.reshape(-1,1)

**#Splitting into train\_set and test\_set**

from sklearn.model\_selection import train\_test\_split X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y, test\_size=0.2, random\_state=0)

**# Training the Simple Linear Regression model on the Training set**

from sklearn.linear\_model import LinearRegression regressor = LinearRegression()

regressor.ﬁt(X\_train, Y\_train)

**# Predicting the Test set results**

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

**#we need to compare y\_predict(by model) and y\_test(data) #Visualizing the training set results**

plt.scatter(x, y, color='red')

plt.plot(x, regressor.predict(x),color='green') plt.title('life expectancy vs happiness score') plt.xlabel('life expectancy') plt.ylabel('happiness')

plt.show()



[FROM THE ABOVE GRAPH AND ABOVE REGRESSION MODEL],WE CAN OBSERVE THAT THERE IS A STRONG LINEAR RELATION BETWEEN lIFE EXPECTANCY AND HAPPINESS INDEX.SO, PROBABLY THE CORRELATION ALSO WILL BE HIGH WHICH IS ALSO PROVED BELOW.

#### #Task-2(Population and happiness)

**#Dividing into x and y**

##### a = dataset1.iloc[:, 2].values b = dataset1.iloc[:, -1].values a=a.reshape(-1,1) b=b.reshape(-1,1)

**#Splitting into train\_set and test\_set**

from sklearn.model\_selection import train\_test\_split A\_train,A\_test,B\_train,B\_test=train\_test\_split(a,b, test\_size=0.2, random\_state=0)

**# Training the Simple Linear Regression model on the Training set**

from sklearn.linear\_model import LinearRegression regressor = LinearRegression()

regressor.ﬁt(A\_train, B\_train)

**# Predicting the Test set results**

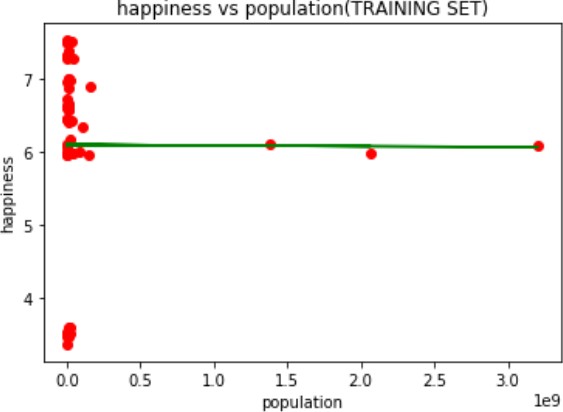
b\_pred = regressor.predict(A\_test)

**#we need to compare y\_predict(by model) and y\_test(data) #Visualizing the training set results**

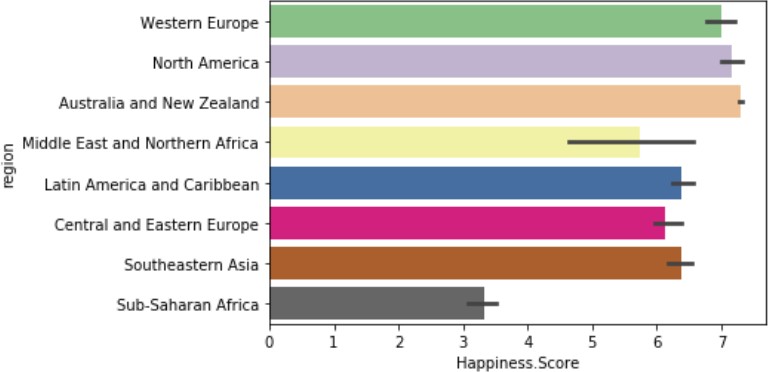
plt.scatter(a, b, color='red')

plt.plot(a, regressor.predict(x),color='green') plt.title('population vs happiness score') plt.xlabel('Population') plt.ylabel('happiness')

plt.show()



[FROM THE ABOVE GRAPH AND ABOVE REGRESSION MODEL],WE CAN OBSERVE THAT THERE IS NO LINEAR RELATION BETWEEN POPULATION AND HAPPINESS INDEX.SO, PROBABLY THE CORRELATION IS ALSO LOW.



[Q2]Do these countries have a lower or higher happiness score?

[A2]Sub-Saharan African region seems to have very less life expectancy when compared with other regions and is also having less happiness index relatively.

[Q3]Is there a relationship between population size and happiness?

[A3]Statistically, we can see that correlation between happiness and population size is quite low [0.01]. This shows that happiness and population are not well related.

[CONCLUSION] [WE HAVE OBSERVED THE FOLLOWING POINTS]

1. Happiness and life expectancy are strongly correlated.
2. Happiness and population are weakly correlated.
3. There are few regions which have very less life expectancy than other regions.
4. So,We understood that if a country improves its happiness index, then its life expectancy can increase.