Suicide Rate Prediction with Machine Learning

Presenting by -

Rameshwari Jadhav

Samarasimha Reddy Baisani

Shubham Deshmukh

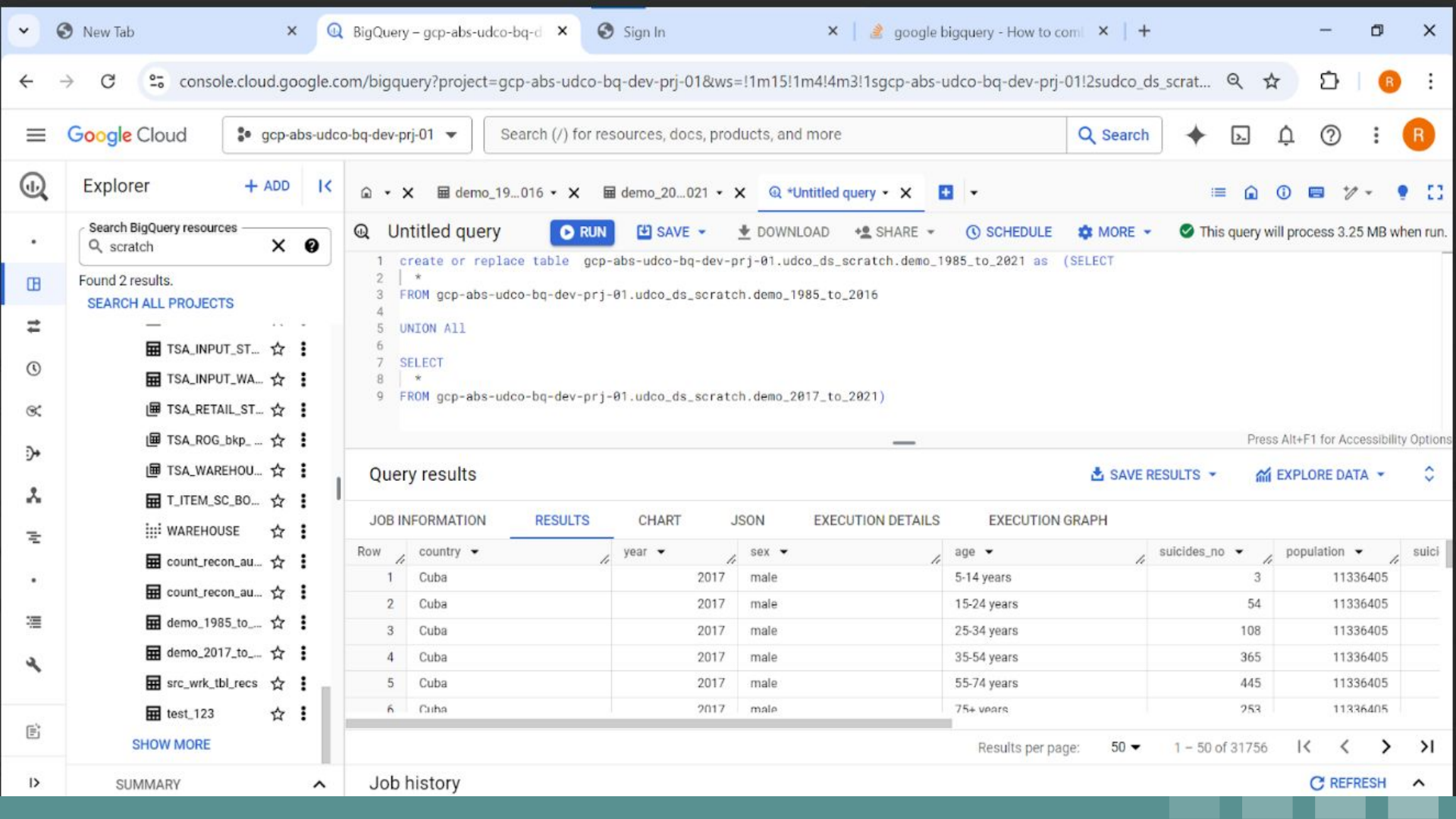
Introduction

- Suicide is a serious public health problem.
- The World Health Organization (WHO) estimates that every year close to 800 000 people take their own life, which is one person every 40 seconds and there are many more people who attempt suicide.
- Suicide occurs throughout the lifespan and was the second leading cause of death among 15-29- year-olds globally in 2016.
- The objective of this project is to predict the suicide rates using Machine Learning algorithms and to analyzing significant patterns features that result in increase of suicide rates globally.
- The project is done on Google Colaboratory

Dataset Details

- The dataset is from Kaggle. This is a compiled dataset pulled from four other datasets linked by time and place from year 1985 to 2016.
- The source of those datasets is WHO, World Bank, UNDP and a dataset published in Kaggle.
- It has 27820 samples and 12 features.
- The features in the dataset are:
  - – country, year,sex, age group, country-year, generation (based on age grouping average).
  - – count of suicides, population,suicide rate, HDI for year, gdp_for_year, gdp_per_capita.
- The number of countries in the data set are 101.

How we merged our data sets ?


New Tab
BigQuery – gcp-abs-udco-bq-d
Sign In
google bigquery - How to com
console.cloud.google.com/bigquery?project=gcp-abs-udco-bq-dev-prj-01&ws=!1m15!1m4!4m3!1sgcp-abs-udco-bq-dev-prj-01!2sudco_ds_scrat...
Google Cloud
gcp-abs-udco-bq-dev-prj-01
Search (/) for resources, docs, products, and more
Search
Explorer
+ ADD
Search BigQuery resources
scratch
Found 2 results.
SEARCH ALL PROJECTS
TSA_INPUT_ST...
TSA_INPUT_WA...
TSA_RETAIL_ST...
TSA_ROG_bkp_...
TSA_WAREHOU...
T_ITEM_SC_BO...
WAREHOUSE
count_recon_au...
count_recon_au...
demo_1985_to_...
demo_2017_to_...
src_wrk_tbl_recs
test_123
SHOW MORE
SUMMARY
demo_19...016
demo_20...021
*Untitled query
Untitled query
RUN
SAVE
DOWNLOAD
SHARE
SCHEDULE
MORE
This query will process 3.25 MB when run.
create or replace table gcp-abs-udco-bq-dev-prj-01.udco_ds_scratch.demo_1985_to_2021 as (SELECT
*
FROM gcp-abs-udco-bq-dev-prj-01.udco_ds_scratch.demo_1985_to_2016
UNION All
SELECT
*
FROM gcp-abs-udco-bq-dev-prj-01.udco_ds_scratch.demo_2017_to_2021)
Press Alt+F1 for Accessibility Options
Query results
SAVE RESULTS
EXPLORE DATA
JOB INFORMATION
RESULTS
CHART
JSON
EXECUTION DETAILS
EXECUTION GRAPH
Row country year sex age suicides_no population suici
1 Cuba 2017 male 5-14 years 3 11336405
2 Cuba 2017 male 15-24 years 54 11336405
3 Cuba 2017 male 25-34 years 108 11336405
4 Cuba 2017 male 35-54 years 365 11336405
5 Cuba 2017 male 55-74 years 445 11336405
6 Cuba 2017 male 75+ years 253 11336405
Results per page: 50
1 – 50 of 31756
Job history
REFRESH

DATA PREPROCESSING:

Null value check. HDI for year has 19456 null values. So dropped the column.

Duplicate column - country-year which is a combination of values in country & year columns. So, the column is dropped.

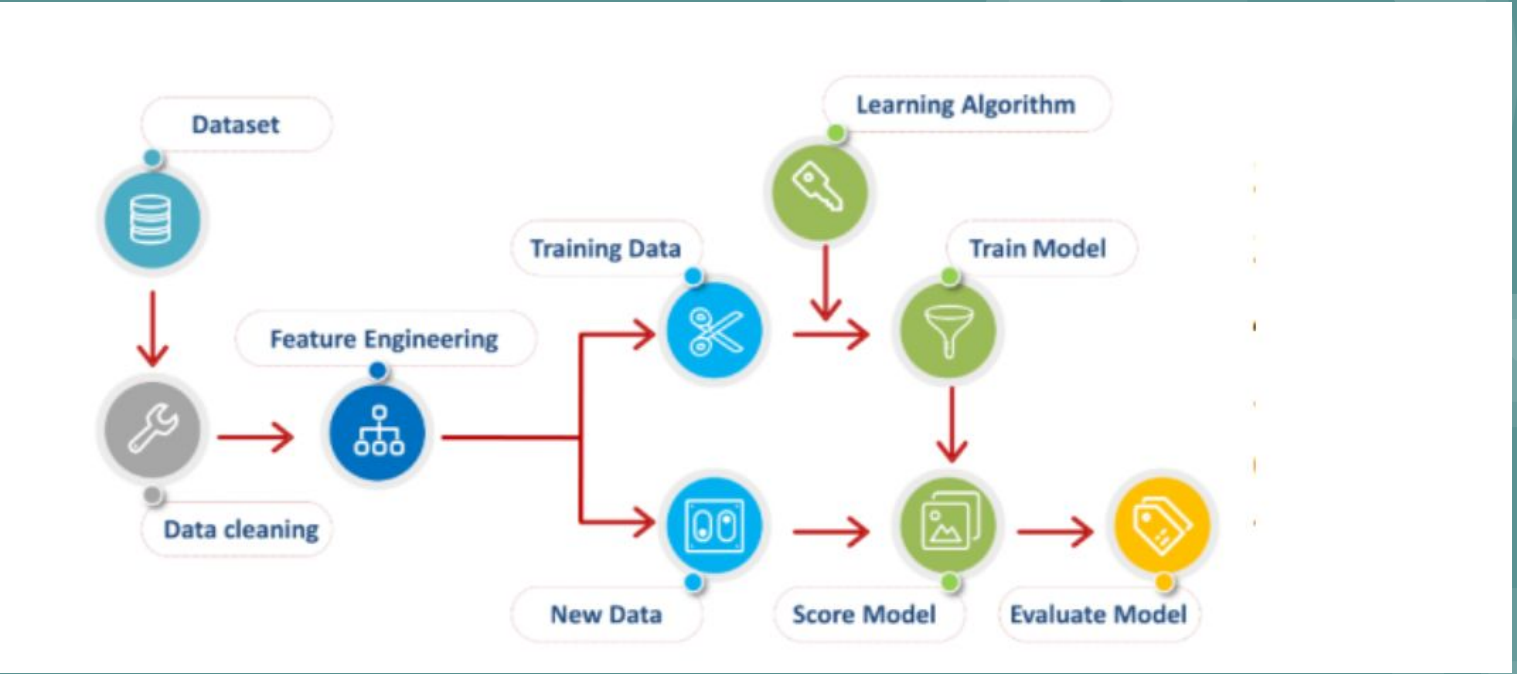
The numerical features of the dataset are scaled using RobustScalar.

- count of suicides, population, suicide rate, gdp_for_year, gdp_per_capita.

The categorical features are encoded by LabelEncoder.

- country, sex, age group, generation

Approach


Dataset
Learning Algorithm
Training Data
Train Model
Feature Engineering
Data cleaning
New Data
Score Model
Evaluate Model

Loading Data

```
#Loading data into dataframe

data = pd.read_csv("suicide_data.csv")
data.head()
```

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	NaN	2,156,624,900	796
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	NaN	2,156,624,900	796
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	NaN	2,156,624,900	796
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	NaN	2,156,624,900	796
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	NaN	2,156,624,900	796

Familiarizing with data

```
#Information about the dataset

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27820 entries, 0 to 27819
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   country         27820 non-null  object
 1   year            27820 non-null  int64
 2   gender          27820 non-null  object
 3   age_group       27820 non-null  object
 4   suicide_count   27820 non-null  int64
 5   population      27820 non-null  int64
 6   suicide_rate    27820 non-null  float64
 7   country-year    27820 non-null  object
 8   HDI for year    8364 non-null   float64
 9   gdp_for_year    27820 non-null  object
 10  gdp_per_capita  27820 non-null  int64
 11  generation      27820 non-null  object
dtypes: float64(2), int64(4), object(6)
memory usage: 2.5+ MB
```

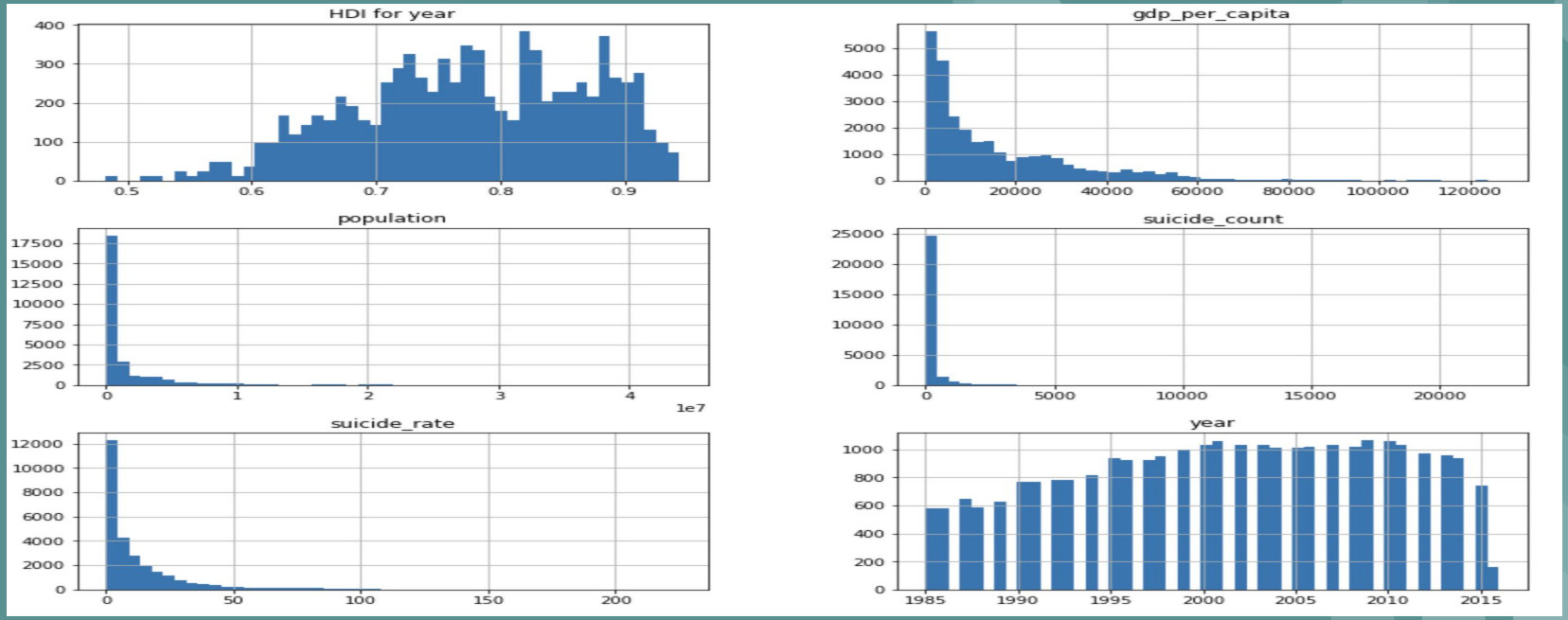
Observations after familiarizing with data

- HDI for year column has missing values. None of the other columns have any missing values. So considering to remove HDI from the dataset.
- The age feature has 6 unique age groups
- Age is grouped into year buckets as categorical format which needs to be encoded.
- Gender should be encoded.
- Scale required numerical features.
- The generation feature has 6 types of generations.
- Generation could be encoded as well.

Visualising the data

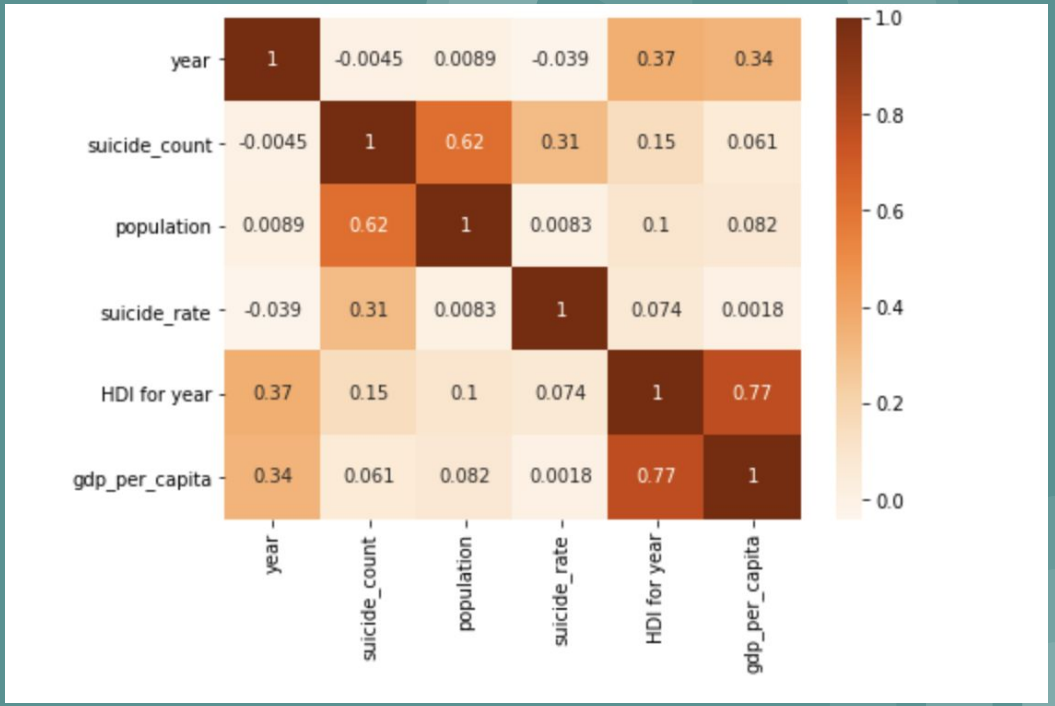
The above bar plot shows that the suicide cases are more in male population.

```
data.hist(bins = 50,figsize = (15,11))
```



Correlation heatmap

Few plots and graphs are displayed to find how the data is distributed and the how features are related to each other.

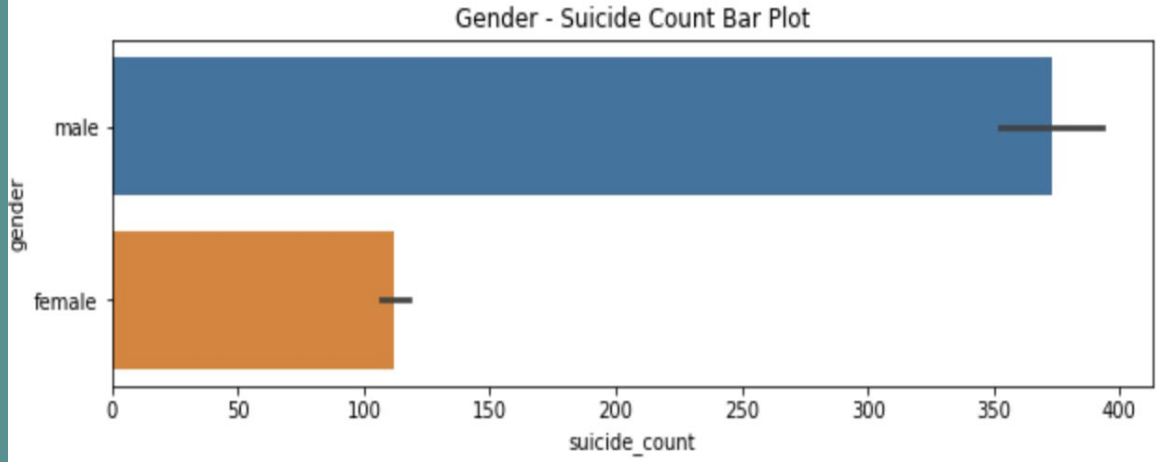


Bar Plot

The above bar plot shows that the suicide cases are more in male population.

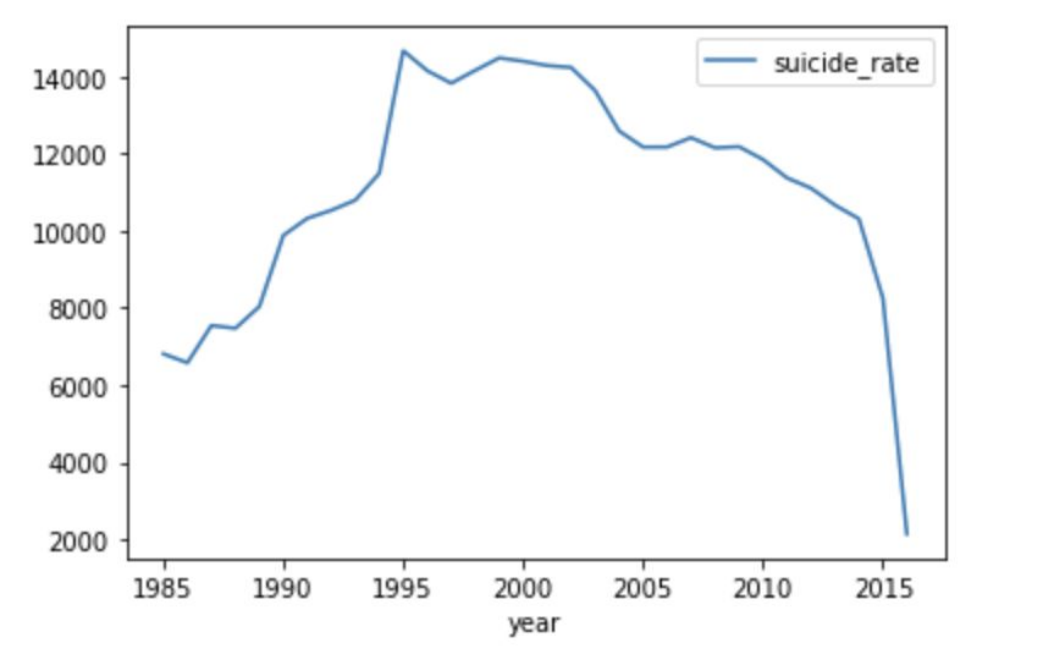
```
#Gender and suicide count bar plot

plt.figure(figsize=(10,3))
sns.barplot(data.suicide_count,data.gender)
plt.title('Gender - Suicide Count Bar Plot')
plt.show()
```



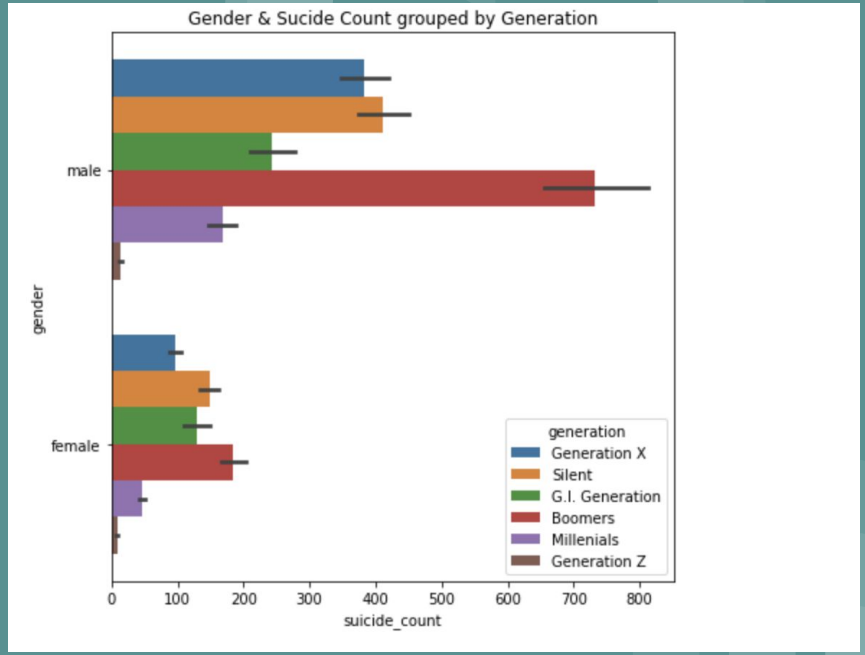
Line plot of year and sucide rate

The observations from the above plot are that the suicide rate had grown rapidly from year 1990 & the rate of suicide has drastically reduced in year 2016. The dataset was collected during early 2016. So all the suicide cases of 2016 are not recorded in the dataset.



Bar plot

- In the case of generation, the Bloomers generation had more suicide cases followed by Silent generation irrespective of the gender.
- Even when considered generation, males are more prone to commit suicide.



Data Preprocessing & EDA

```
data.describe()
```

	year	suicide_count	population	suicide_rate	HDI for year	gdp_per_capita
count	27820.000000	27820.000000	2.782000e+04	27820.000000	8364.000000	27820.000000
mean	2001.258375	242.574407	1.844794e+06	12.816097	0.776601	16866.464414
std	8.469055	902.047917	3.911779e+06	18.961511	0.093367	18887.576472
min	1985.000000	0.000000	2.780000e+02	0.000000	0.483000	251.000000
25%	1995.000000	3.000000	9.749850e+04	0.920000	0.713000	3447.000000
50%	2002.000000	25.000000	4.301500e+05	5.990000	0.779000	9372.000000
75%	2008.000000	131.000000	1.486143e+06	16.620000	0.855000	24874.000000
max	2016.000000	22338.000000	4.380521e+07	224.970000	0.944000	126352.000000

```
#checking the data for null or missing values

data.isnull().sum()
```

```
#dropping the HDI for year column

data = data.drop(['HDI for year'], axis = 1)
data.shape
```

```
#dropping the country-year for year column

data = data.drop(['country-year'], axis = 1)
data.shape
```

```
#encoding the categorical features with LabelEncoder

from sklearn.preprocessing import LabelEncoder
categorical = ['country', 'year','age_group', 'gender', 'generation']
le = sklearn.preprocessing.LabelEncoder()

for column in categorical:
    data[column] = le.fit_transform(data[column])
```

```
#creating a copy of dataset for statistical test

stat_data = data.copy()
stat_data
```

country	year	gender	age_group	suicide_count	population	suicide_rate	gdp_for_year	gdp_per_capita	generation
0	2	1	0	21	312900	6.71	2,156,624,900	796	2
0	2	1	2	16	308000	5.19	2,156,624,900	796	5
0	2	0	0	14	289700	4.83	2,156,624,900	796	2
0	2	1	5	1	21800	4.59	2,156,624,900	796	1
0	2	1	1	9	274300	3.28	2,156,624,900	796	0
...	...	...	...	...	...	...	...	...	...
100	29	0	2	107	3620833	2.96	63,067,077,179	2309	2
100	29	0	5	9	348465	2.58	63,067,077,179	2309	5
100	29	1	3	60	2762158	2.17	63,067,077,179	2309	3
100	29	0	3	44	2631600	1.67	63,067,077,179	2309	3
100	29	0	4	21	1438935	1.46	63,067,077,179	2309	0

Splitting the Data

```
# Sepratating & assigning features and target columns to X & y

y = data['suicide_rate']
X = data.drop('suicide_rate',axis=1)
X.shape, y.shape
```

```
((27820, 9), (27820,))
```

```
# Splitting the dataset into train and test sets: 80-20 split

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 = 12)
X_train.shape, X_test.shape
```

```
((22256, 9), (5564, 9))
```

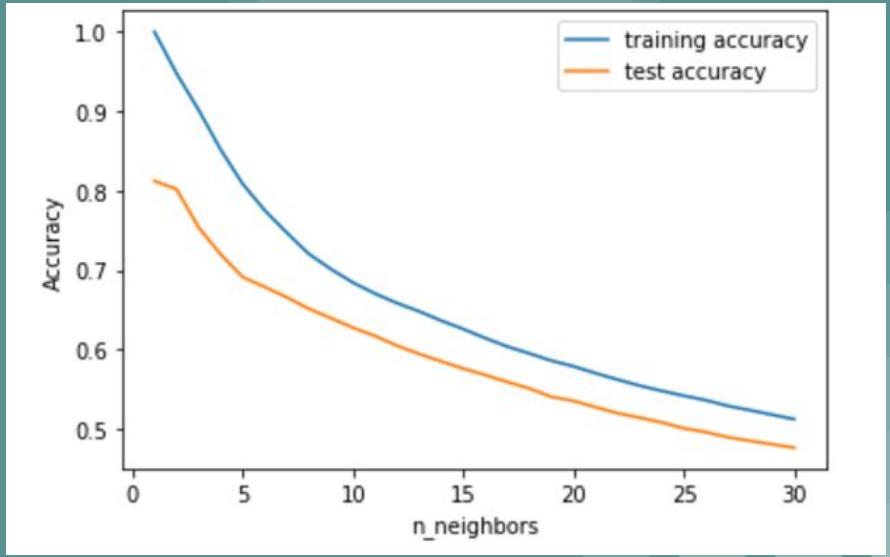
Model Building & Training

Compare performance of various regression models

- Linear Regression.
- Decision Trees.
- Random Forest.
- Gradient Boosting.
- XGBoost.
- Multilayer Perceptrons (MLPs).
- Custom Ensemble: SuperLearner.

k-Nearest Neighbors Regression

- A simple and effective machine learning algorithm.
- Predicts the target value by averaging the values of the k-nearest data points.
- Relies on distance metrics (e.g., Euclidean distance) to find nearest neighbors.

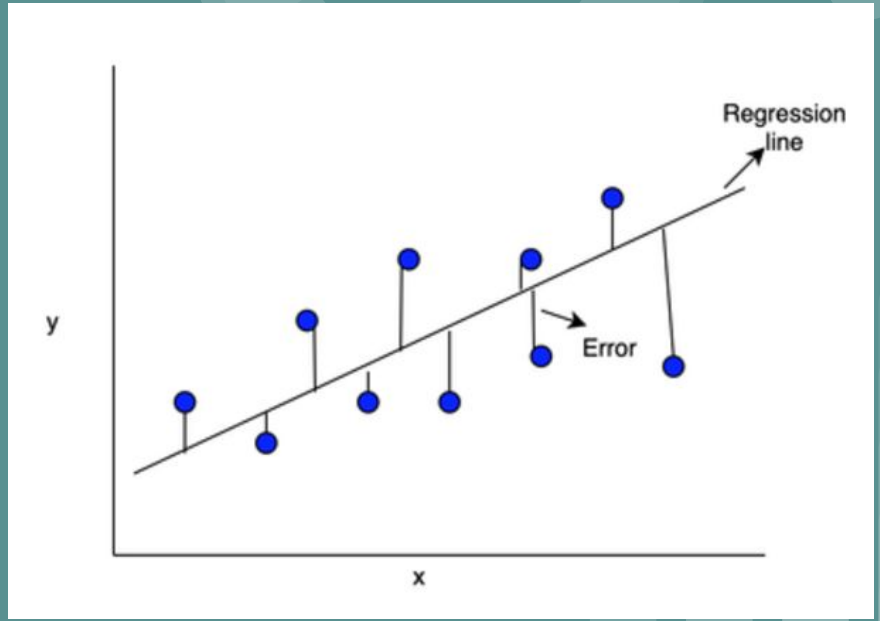


Linear Regression

What is Linear Regression?

- A fundamental and simple regression model.
- Predicts the target variable by fitting a linear relationship between input features and the target.
- Equation: **y=wX+by = wX + by=wX+b**

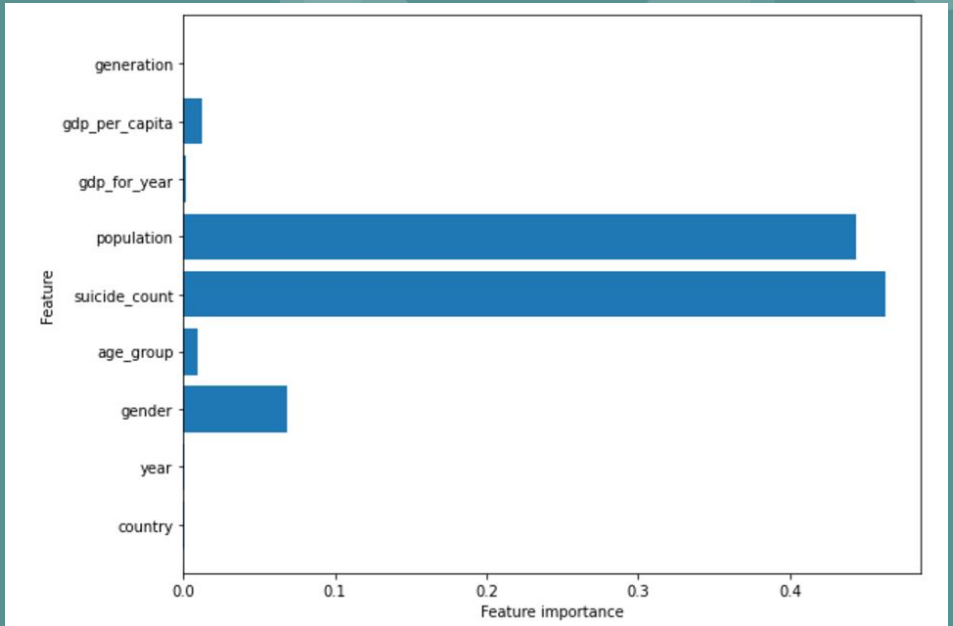
Where www is the weight (coefficient), XXX is the input feature, and bbb is the bias (intercept).



Regression

What is Decision Trees Regression?

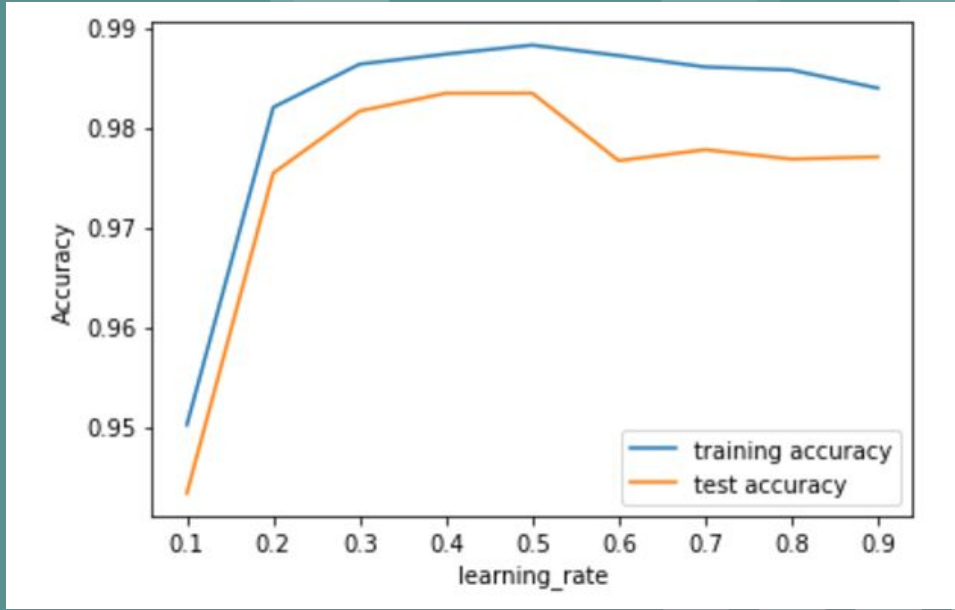
- A supervised learning model that uses a tree-like structure to make decisions.
- Predicts the target variable by learning a hierarchy of decision rules (if/else conditions).
- Splits data recursively based on feature values to minimize error.



Gradient Boosted Regression Trees

What is Gradient Boosted Regression Trees?

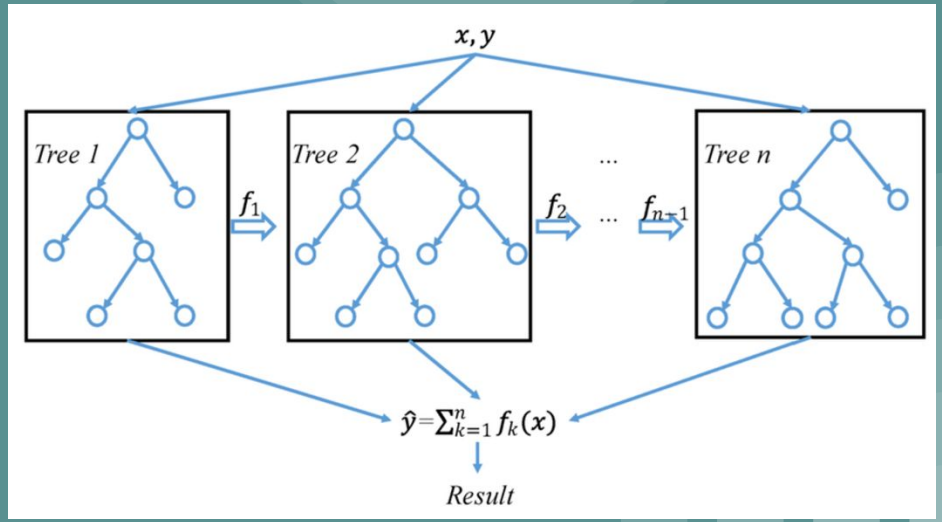
- An ensemble machine learning method that builds trees sequentially.
- Each tree corrects errors made by previous trees, optimizing a loss function.



XG Boost

What is XGBoost?

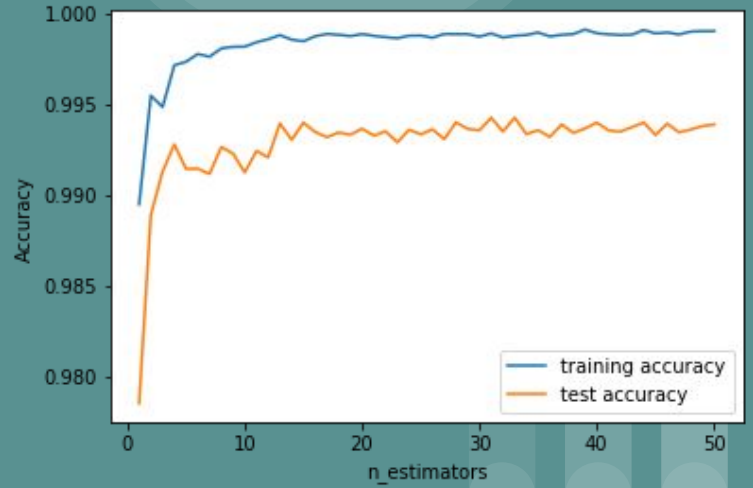
- **XGBoost (eXtreme Gradient Boosting)** is a scalable and efficient implementation of Gradient Boosted Decision Trees.
- Optimized for speed and performance using parallel processing and regularization.
- Handles missing data and overcomes overfitting effectively.



Bagging Regression

What is Bagging Regressor?

- **The Bagging Regressor** is an ensemble estimator that uses the voting or averaging approach to aggregate the individual predictions of each random subset of the Train dataset after fitting the base estimator to each one. Decision trees serve as the base estimator in this case.



Custom Ensemble - SuperLearner:

ML-Ensemble (mlens) is a Python library designed to facilitate the creation of custom ensemble learning models. It provides an efficient framework for building ensembles with a feed-forward network structure, enabling stacking and blending strategies while maintaining compatibility with scikit-learn.

Key Features

1. **Scikit-learn Compatible**:
   - Fully compatible with the scikit-learn API, making it easy to integrate with existing pipelines.
2. **Feed-Forward Network**:
   - Ensembles are constructed as layers of models (base learners) stacked on top of each other.
   - Each layer uses predictions from the previous layer as input.
3. **Layered Structure**:
   - Multiple models can be grouped in layers.
   - You can combine models of different types and adjust their hyperparameters independently.

Advantages

- **Custom Architectures**: Create highly tailored ensembles to suit specific datasets.
- **Improved Performance**: Combines multiple models to reduce bias, variance, or both.
- **Ease of Use**: Simplifies the creation of complex stacking and blending ensembles.
- **Efficient Handling of Large Datasets**: Optimized for scalability and memory usage.

Limitations

- **Complexity**: Designing custom ensembles requires careful tuning of layers and models.
- **Runtime Overhead**: Ensembles with many models can be computationally expensive despite parallelization.

Comparison of Models:

The models' performance is compared using a dataframe. This dataframe's columns are the lists made to hold the model's output.

```
#creating dataframe
results = pd.DataFrame({ 'ML Model': ML_Model,
    'Train Accuracy': acc_train,
    'Test Accuracy': acc_test,
    'Train RMSE': rmse_train,
    'Test RMSE': rmse_test})
```

Results

	ML Model	Train Accuracy	Test Accuracy	Train RMSE	Test RMSE
0	k-Nearest Neighbors Regression	1.000	0.812	0.000	0.536
1	Linear Regression	0.288	0.296	1.013	1.037
2	Decision Tree	0.967	0.952	0.220	0.272
3	Random Forest	0.987	0.980	0.137	0.176
4	Gradient Boosted Regression	0.988	0.983	0.130	0.159
5	Multilayer Perceptron Regression	0.926	0.928	0.326	0.331
6	XGBoost Regression	0.993	0.988	0.100	0.134
7	Bagging Regression	0.994	0.982	0.096	0.166
8	Ensemble_SuperLearner	0.912	0.910	0.357	0.371

```
#Sorting the datafram on accuracy
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

	ML Model	Train Accuracy	Test Accuracy	Train RMSE	Test RMSE
6	XGBoost Regression	0.993	0.988	0.100	0.134
4	Gradient Boosted Regression	0.988	0.983	0.130	0.159
7	Bagging Regression	0.994	0.982	0.096	0.166
3	Random Forest	0.987	0.980	0.137	0.176
2	Decision Tree	0.967	0.952	0.220	0.272
5	Multilayer Perceptron Regression	0.926	0.928	0.326	0.331
8	Ensemble_SuperLearner	0.912	0.910	0.357	0.371
0	k-Nearest Neighbors Regression	1.000	0.812	0.000	0.536
1	Linear Regression	0.288	0.296	1.013	1.037

Statistical Tests:

Statistical tests are used in hypothesis testing. They can be used to:

- Determine whether a predictor variable has a statistically significant relationship with an outcome variable.
- estimate the difference between two or more groups

```
#improting required libraries
from scipy import stats
```

Test 1: To check the difference in suicide rates between male and female

Using independent sample t-test to check the difference in suicide rates between male and female. The hypothesis statements for this test are:

H0: There is no difference in the suicide rates among male and female (Null).
H1: There is difference in the suicide rates among male and female (Alternate).

```
#collecting male suicide rate data
male = stat_data['suicide_rate'][stat_data['gender'] == 1]
male
```

```
#collecting female suicide rate data
female = stat_data['suicide_rate'][stat_data['gender'] == 0]
female
```

```
#calculating p value
ttest,pval = stats.ttest_rel(male, female)

if pval<0.05:
    print("Reject null hypothesis")
else:
    print("Accept null hypothesis")
```

Test Conclusion: By performing T-test, the result obtained is to reject the null hypothesis. This basically means that there is different in suicide rates of male & female.

Test 2: To find out the dependence of suicide rate on the age.

Finding out whether there is a dependence of suicide rate on the age using the Chi-Square test. The hypothesis statements for this test are:

H0: Suicide rate and age are independent (Null).
H1: Suicide rate and age are dependent (Alternate).

```
#Creating Contingency Table
contingency_table = pd.crosstab(stat_data.suicide_rate, stat_data.age_group)
```

```
#Significance Level 5%
alpha=0.05
```

```
#compare chi_square_statistic with critical_value and p-value which is the
#probability of getting chi-square>0.09 (chi_square_statistic)
if chistat>=critical_value:
    print("Reject H0,There is a dependency between Age group & Suicide rate.")
else:
    print("Retain H0,There is no relationship between Age group & Suicide rate.")

if p<=alpha:
    print("Reject H0,There is a dependency between Age group & Suicide rate.")
else:
    print("Retain H0,There is no relationship between Age group & Suicide rate.")
```

Test Conclusion: The null hypothesis is rejected as a consequence of the Chi-Square test. This essentially indicates a relationship between age group and suicide rate.

Conclusion:

The project provided valuable insights into the application of various machine learning models, highlighting their performance differences and the impact of parameter tuning on model optimization. By working through this notebook, I gained a deeper understanding of how to adjust parameters like learning rates and tree depths to enhance model accuracy and efficiency. Analyzing the suicide dataset revealed a consistent and concerning trend: across all age groups and generations, males are significantly more prone to suicide than females. This finding underscores the need to address systemic and societal factors contributing to this disparity, offering a meaningful application of data science to real-world issues.

THANK YOU

ANY QUESTIONS?