Machine Learning Engineer Nanodegree
Capstone Project Review

By,
Aakar Mutha

I. Definition

Domain Addressed

Suicide is one of the major causes of deaths in the world. It is estimated that around 20 Million people attempt to commit suicide every year. People commit suicide because of reasons like Mental disorders, including depression, bipolar disorder and many more. The people who are more prone to commit suicide are in the age groups of 15 – 30 years and 70 + years. The dataset I am using is posted in Kaggle by the World Health Organization.

The motivation for this project was to combat this problem and to predict approximately the number of suicides that can take place in a given country in a given year according to the historical data. This prediction will help the government as well as the NGOs to spread awareness.

Problem Statement

The goal is to create a machine learning model which will give us the number of suicides that occur in a particular age group in a country. This model will help the government and NGOs to better organise campaigns for suicide prevention.

This will also help to organize campaigns that are targeted to a particular age group.

Solution Statement

The solution will utilize the fact that the historical data can be used correlate and predict the number of suicides that may occur. This will be helpful for us to use supervised learning models and predict the suicidal behaviour of people in the coming years. I will be using the 5 columns namely country, year, sex, age and population as the input to the model. I will be testing various regression techniques like SVR, Linear Regression, Polynomial Regression, Decision Trees, etc.

I will also evaluate other bagging and boosting models to determine the best model. I will also compare all the results obtained from all the models.

Evaluation Matrix

For the evaluation of the models, I will be using the r^2 score as the evaluation factor. The goal is to increase the r^2 value along with reducing the MSE as well as RMSE values.

I will also be usnig the Kfold cross validation score to check for overfitting and underfitting.

II. Analysis

Dataset Exploration

The dataset is provided by Szamil and is named as WHO Suicide Statistics. It has 43,776 entries in each of its 6 columns.
https://www.kaggle.com/szamil/who-suicide-statistics

Variable	Description
country	Name of the country from where the data was recorded
year	Year in which the data was recorded
sex	Gender of the data
age	Age Group in which the data falls
suicide_no	Number of suicides occered in that age group in that year
population	population of the country in that year in the specified age group

https://docs.google.com/spreadsheets/d/1px2CKqXjjgkzWolwIgsf-QKvG0IfSAI4H6TqLf3NTbU/edit?usp=sharing

The input to the model will be country, year, sex, age and population. This will all be passed through Label encoder so that it is converted to categorical variables.

The dataset consists 141 different countries.

The dataset is for 38 years (1979 – 2016) of historical data.

The dataset is divided in 6 age groups.

Let's start by checking the dataset for null values. Since our dataset is large we can remove the rows having null values.

```
# Checking for the total data points
data.shape
```

```
(43776, 6)
```

```
[7] # Checking for any null values
    data.isnull().sum()
```

```
country          0
year             0
sex              0
age              0
suicides_no    2256
population     5460
dtype: int64
```

After dropping all the null values

```
[13] data.dropna(inplace = True)
     data.isnull().sum()
```

```
country        0
year           0
sex            0
age            0
suicides_no    0
population     0
dtype: int64
```

```
[14] # Checking for the total data points
     # after dropping the null rows
     data.shape
```

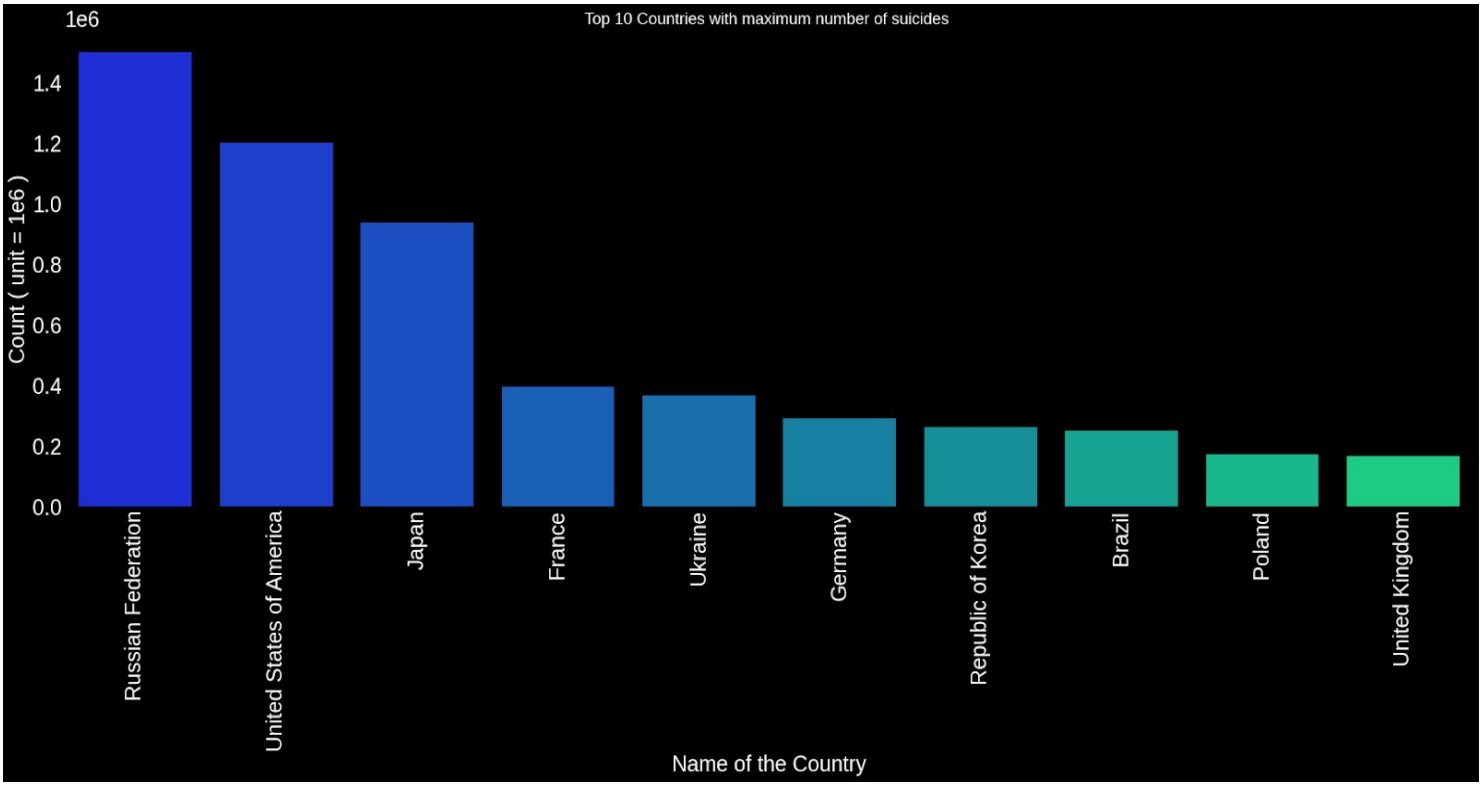
```
(36060, 6)
```

Checking the info of the dataframe after dropping the rows

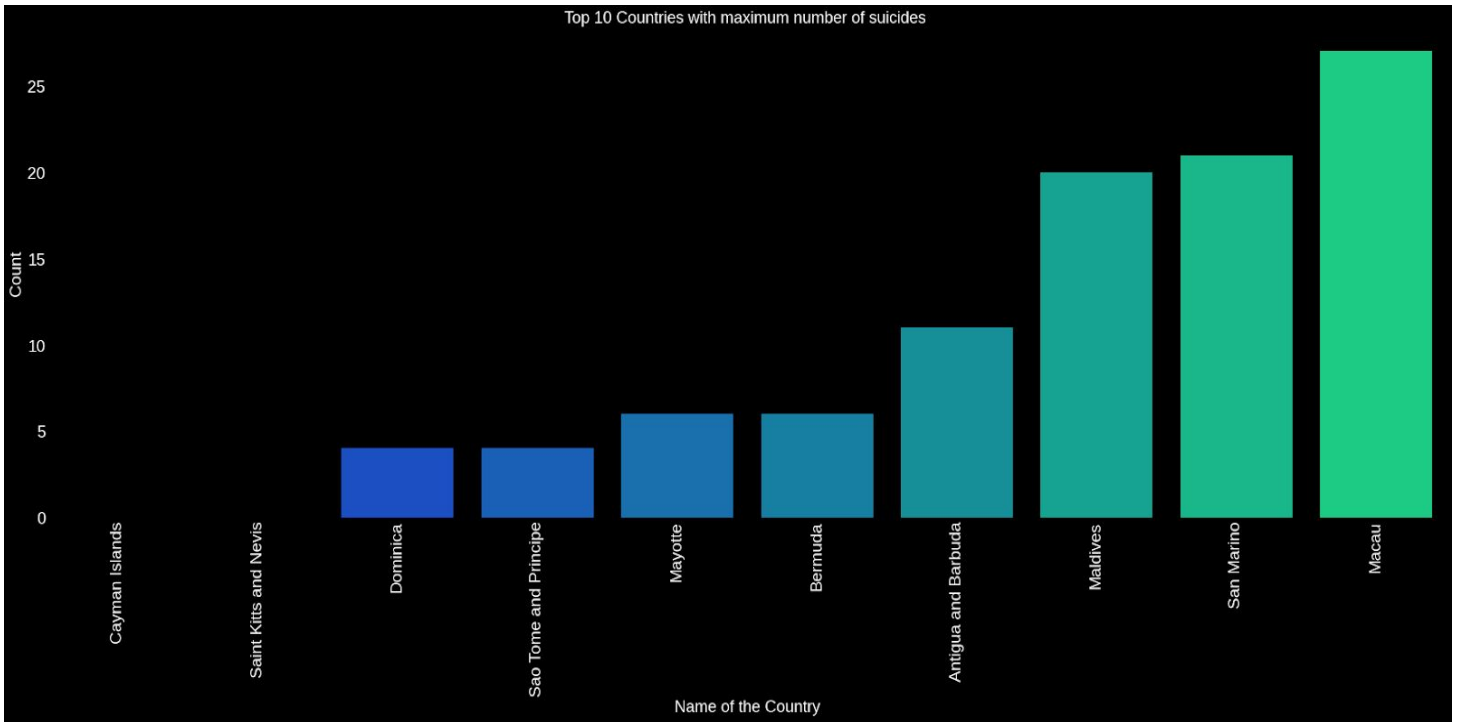
```
[16] data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 36060 entries, 6204 to 25858
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   country      36060 non-null  object 
 1   year         36060 non-null  int64  
 2   sex          36060 non-null  object 
 3   age          36060 non-null  object 
 4   suicides_no  36060 non-null  float64
 5   population   36060 non-null  float64
dtypes: float64(2), int64(1), object(3)
memory usage: 1.9+ MB
```

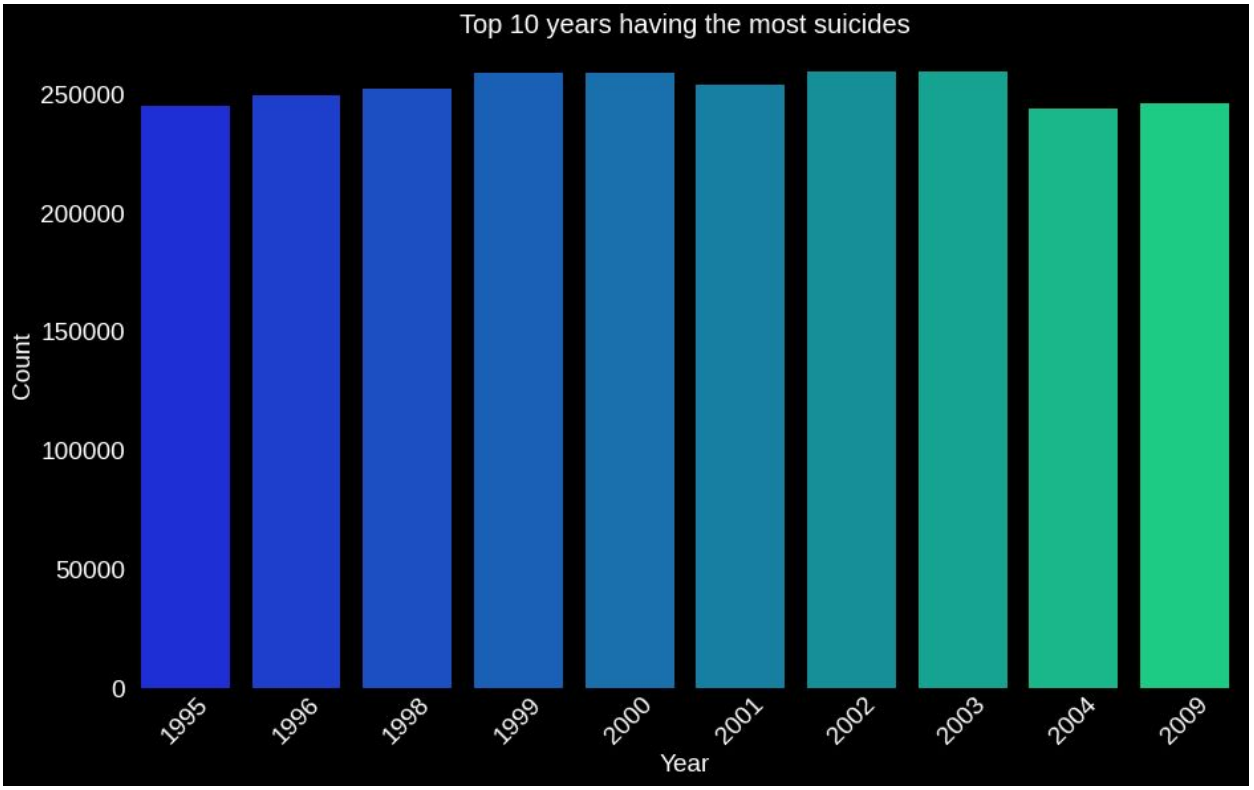
Top 10 countries having the most suicides



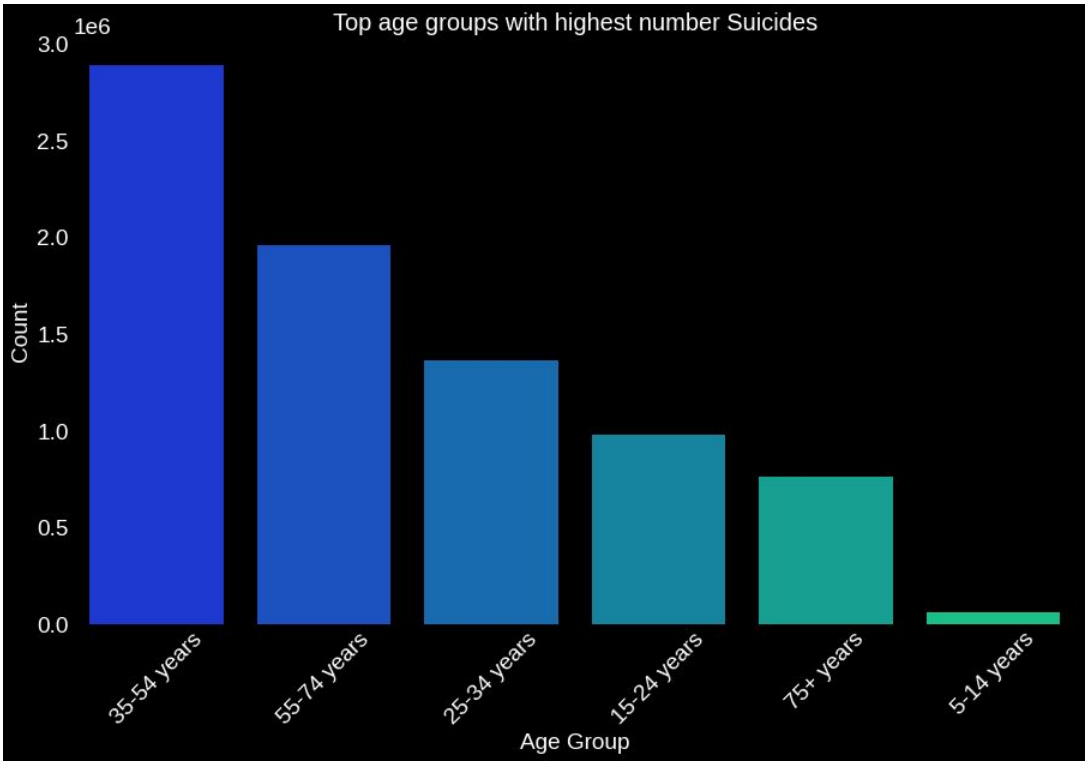
Top 10 countries with the least number of suicides



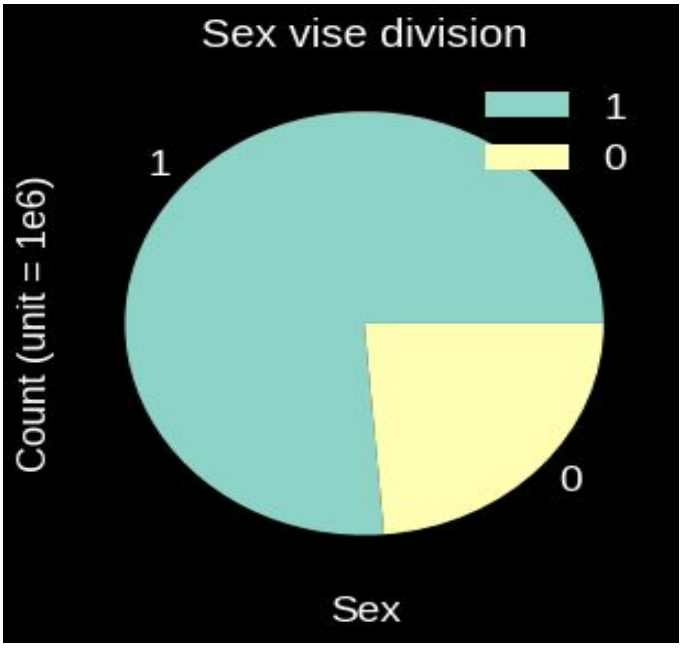
Top 10 Years having the greatest number of suicides



Distribution of number of suicides over the different Age Groups



Sex wise distribution of suicides



1. Male 2. Female

From the above representations we can see that the most number od suicides are committed in the Russian Federation. On the other hand, the least number of suicides are committed in the Cayman Islands. Most people committing suicide are in the age group of 35-54 years and males are more prone to commit suicide.

Algorithms and Techniques

The above problem is a regression problem. So to implement the solution we have a choice of three types of algorithms.

1. Linear algorithms like linear regression, Losso, Elastic Net, etc.
2. Boosting algorithms like Ada boost, Gradient boosting, etc
3. Bagging algorithms like Random Forest, Extra trees, etc

The linear models like Losso, Linear regression are very good at problems which have few number of class variables. This is beacaue, it is easier to fit a line through it.

Boosting is an iterative technique which adjusts the weight of an observation based on the last classification. If an observation was incorrectly prdicted, it tries to increase the weight of this observation.

Bagging is a way to decrease the variance in the prediction by generating additional data for training from the dataset using combinations with repetitions to produce multi-sets of the original data. **Since, we are dealing with a large dataset and a data having great variance, the bagging algorithms were the go to choice for this project.**

So, to get the best algorithm and to compare the performance of the different linear, boosting and bagging algorithms, I decided to implement the algorithms. The performance of each algorithm is compared by the MSE, RMSE and r^2 scores.

The algorithms used were all using their default parameters. I decided to do hyperparameter tuning after I had found the best algorithm among the bagging ones.

Benchmark Model

My benchmark model is Decision Tree Regression. This model got

MSE : 192686.9692526345, RMSE : 438.96123889545703 ,
r2_score : 0.7462256538832388

My goal is to improve the r2 score along with decreasing the MSE and RMSE. I choose the decision tree as the benchmark model since, it is one of the most basic bagging regression model. So, it forms a good benchmark model.

Evaluation Matrix

The goal is to maximize the r2 score and minimize the MSE and RMSE values. This will allow us to account the variability in the data. Minimizing the RMSE and MSE will allow us to know how well our model has performed.

I have also used the KFold cross validation method to check for overfitting.

III. Methodology

Data pre-processing

```
# Checking for any null values
data.isnull().sum()
```

```
country           0
year              0
sex               0
age               0
suicides_no    2256
population     5460
dtype: int64
```

```
# Checking for the total data points
data.shape
```

```
(43776, 6)
```

It was found that the dataset is of 141 countries

```
print("Number of unique countries :" , data['country'].nunique())
```

```
Number of unique countries : 141
```

Since the data is large, it was possible to remove the rows having any null values. After dropping the null values , there were 36060 rows and 6 columns in the dataset.

I also found that over the years, 6104173 males committed suicide and 1894294 females committed suicide.

To convert these string features into numerical data, the sex column was passed through label encoder.

The country and age columns were passed through the get_dummies function to encode each individual country and age group in a different column. This ends the preprocessing. The first column of this encoded data is dropped to avoid the dummy variable trap.

After the preprocessing the data has shape of (36060,126)

```
data.shape
```

```
(36060, 126)
```

All the countries have been converted to individual columns.

Implementation

To start implementing the various algorithms, the data is then split into training and testing dataset.

```
print("Shapes of train data :")
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

print("Shapes of benchmark data :")
print(bm_x_train.shape)
print(bm_x_test.shape)
print(bm_y_train.shape)
print(bm_y_test.shape)
```

```
Shapes of train data :
(28848, 125)
(7212, 125)
(28848,)
(7212,)
Shapes of benchmark data :
(28848, 4)
(7212, 4)
(28848,)
(7212,)
```

The benchmark dataframe does not have the country column and the age is label encoded.

The train and the test sets are passed through the min max scaler to perform data normalization.

Each algorithm is then declared and with the default hyperparameters from scikit learn. In order to compare the different bagging, boosting and linear algorithms all the algorithms have been implemented.

Refinement

To refine the algorithms implemented I planned to do hyperparameter tuning. Extra Trees Regressor gave me the best performance on the dataset. In order to improve the performance of this algorithm, I decided to test the algorithm by increasing the number of estimators and the number of jobs. I found the best performance was achieved by the algorithm with n_estimators = 115 and n_jobs = 10.

This reduced the MSE error from `3971.087770` to `3830.393082` while also reducing the RMSE and increasing the r^2 score.

IV. Results

The results from implementing the different bagging, boosting and linear algorithms show that the Extra Trees Regressor performs the best without any hyperparameter tuning.

	model_name	MSE	RMSE	r2_score
8	Extra Trees Regressor	3971.087770	63.016567	0.994770
5	Random Forest Regressor	6786.241056	82.378644	0.991062
12	XGBoost Regressor	8641.591981	92.960163	0.988619
10	Bagging Regressor	9774.081811	98.863956	0.987127
11	Desicion Tree Regressor	9895.576539	99.476512	0.986967
9	Histogram Gradient Boosting Regressor	18416.873338	135.708781	0.975744
7	Gradient Boosting Regressor	69556.227819	263.735147	0.908392
6	Ada Boost Regressor	180839.517742	425.252299	0.761829
0	Linear Regression	323457.467427	568.733213	0.573997
4	Ridge	323724.041403	568.967522	0.573646
3	SGD Regressor	326719.472073	571.593800	0.569701
1	Losso	331537.805845	575.793197	0.563355
2	Elastic Net	721017.491108	849.127488	0.050399

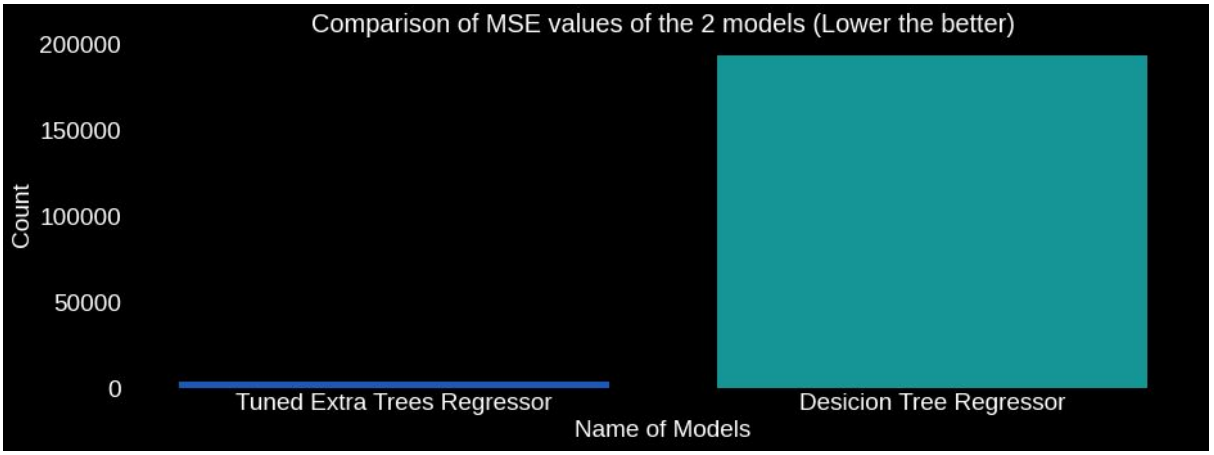
After hyperparameter tuning, the model got

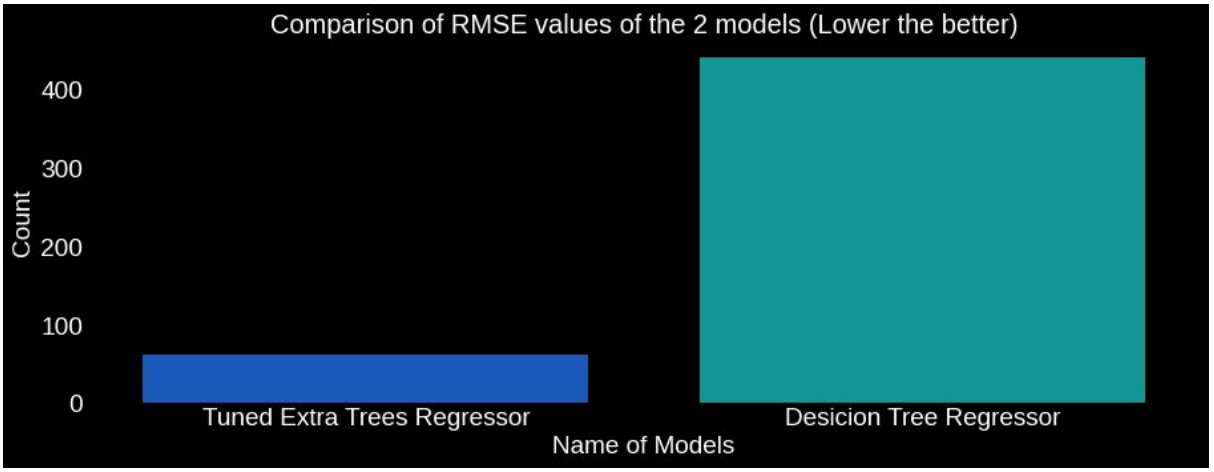
r^2 score = 0.995 , MSE = 3830.393 and RMSE = 61.89

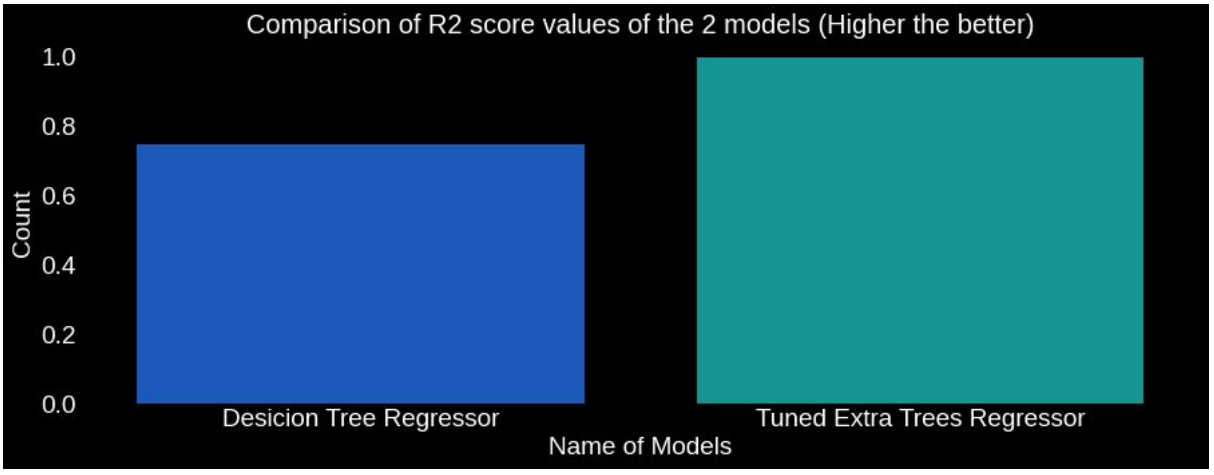
The model got 0.981 as the KFold cross validation score.

Comparing the model with the benchmark

	model_name	MSE	RMSE	r2_score
0	Desicion Tree Regressor	192686.969253	438.961239	0.746226
1	Tuned Extra Trees Regressor	3830.393082	61.890170	0.994955







The Tuned Extra Trees model got 24% more r2 score. It got a lower Mean square of 3830.4 as well as lower Root mean square error.

V. Conclusions

Bagging Regressors perform better with data having a large variance between the classes.

The Extra Trees Regressor Model has better performance in all the evaluation matrix namely MSE, RMSE and r^2 score.

Encoding the string values using label encoder and get_dummies are very important steps. These steps can have a significant impact on the model's performance.

A model with only label encoded classes performs worse than a model with series encoded with dummy variables.

Normalization of the data is also an important step to get better performance from any model.