TTL Lab Presentation

Diamonds Price Analysis and Prediction

3 April 2024

Problem Statement

The problem statement for diamond price analysis and prediction involves conducting a data visualization analysis on the given dataset and building a model that precisely predicts the price of diamonds, considering various features such as carat weight, cut, color, clarity, and dimensions. The objective is to utilize machine learning algorithms to examine historical data and uncover patterns that impact diamond prices.. By understanding these factors, the model aims to provide insights into market trends and assist in pricing decisions for buyers and sellers in the diamond industry.

Data Source and Schema

There are 9 independent variables: Carat, Cut, Color, Clarity, Depth, Table, x, y, and z.

Carat: carat refers to the unique unit of weight measurement used exclusively to weigh gemstones and Diamonds.
Cut: quality of diamond cut.
Color: The color of the diamond.
Clarity: Diamond clarity is a measure of the purity and rarity of the stone, Graded by the visibility of these characteristics under 10-power magnification.
Depth: the depth of diamond is its Height (in millimeters) measured from the Culet (bottom tip) to the table (flat, top surface).
Table: diamond's table is the facet which can be seen when the stone is viewed face up.

x: Diamond X dimension.
y: Diamond Y dimension.
z: Diamond Z dimension.
Target/Dependent Variable: Price of diamonds

Exploratory Data Analysis

What is it?

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

The project we have worked on……
This project attempts to understand various trends present in the Diamonds Dataset and also tries to make anattempt to make a Linear Regression Model which tries to predict the price of any diamond based on itsattributes like carat,cut,color,clarity,etc.
For this project, we shall be using the pre-existing dataset already present in the Seaborn library which provides us with a datset for prices and all other attributes of 53,840 diamonds in total. Some sampling may also be performed in order to derive insights from the entire dataset and to understand various trends exhibitedby the data. This is because performing operations on 53,840 tuples of data may increase the noise and the duration of execution of the programs as well.

```
In [5]: 1 diamonds[diamonds.isnull().any(axis=1)]
```

Out[5]:

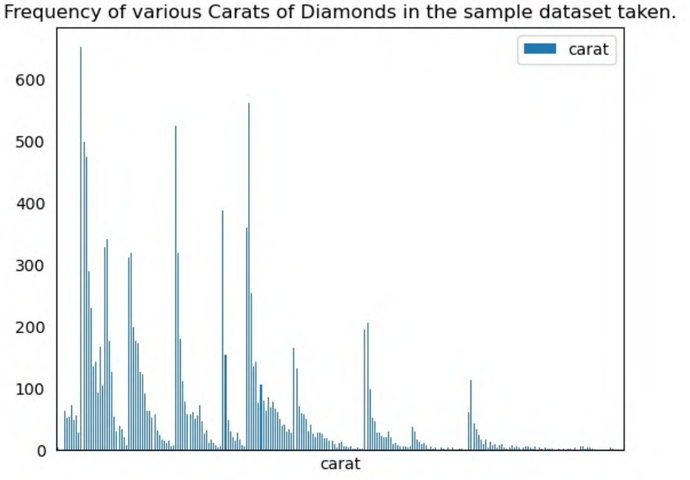
carat	cut	color	clarity	depth	table	price	x	y	z

```
In [6]: 1 diamonds.describe()
```

Out[6]:

	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

```
new_diamonds.groupby('carat')[['carat']].count().plot(kind='bar')
plt.title("Frequency of various Carats of Diamonds in the sample dataset taken.")
plt.tick_params(left = False, right = False , labelleft = True ,
                labelbottom = False, bottom = False)
plt.show()
```




```
new_diamonds.groupby('carat')[['carat']].count().plot(kind='bar')
plt.title("Frequency of various Carats of Diamonds in the sample dataset taken.")
plt.show()
```

Code of Cut vs. Price wrt Mean & Median :-

```
diamonds.groupby('cut')[['price']].median().plot(kind="bar")
plt.grid(linestyle="--")
plt.ylabel("Price")
plt.title("Cut VS Price (With respect to Median)")
plt.show()
diamonds.groupby('cut')[['price']].mean().plot(kind="bar")
plt.grid(linestyle="--")
plt.ylabel("Price")
plt.title("Cut VS Price (With respect to Mean)")
plt.show()
```

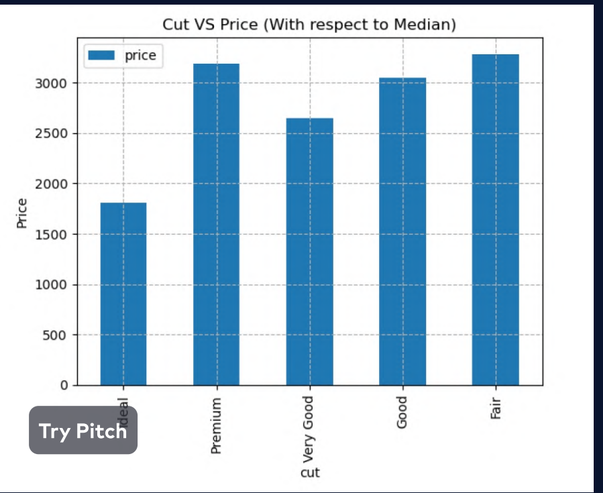
Code of Price vs. Color graph :-

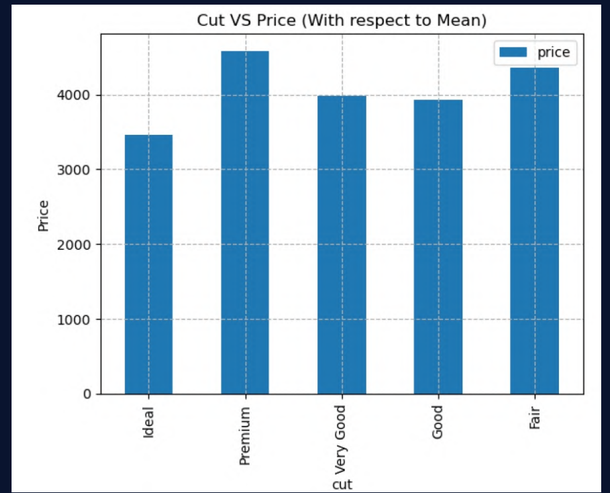
```
new_diamonds.groupby("color")[["price"]].median().plot(kind="bar")
plt.ylabel("price")
plt.grid(linestyle='--')
plt.show()
```

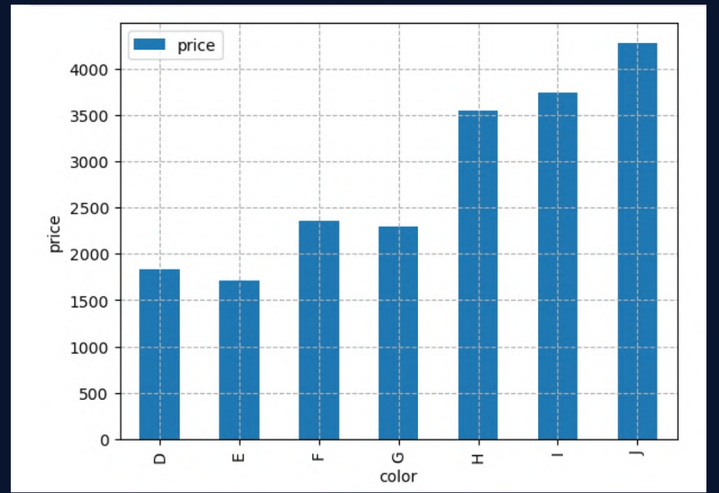
Code of Price vs. Clarity graph :-

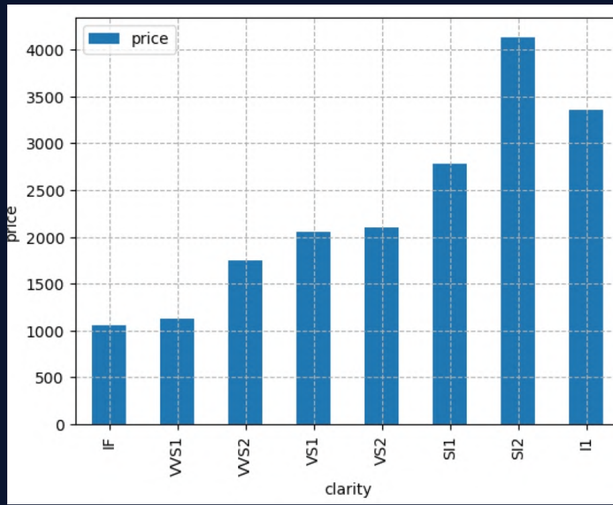
```
new_diamonds.groupby("clarity")[["price"]].median().plot(kind="bar")
plt.ylabel("price")
plt.grid(linestyle='--')
plt.show()
```

Below are few bar-graphs given
for the comparative study of diamond price analysis. We analyse the dataset & code accordingly to summarize the stats.







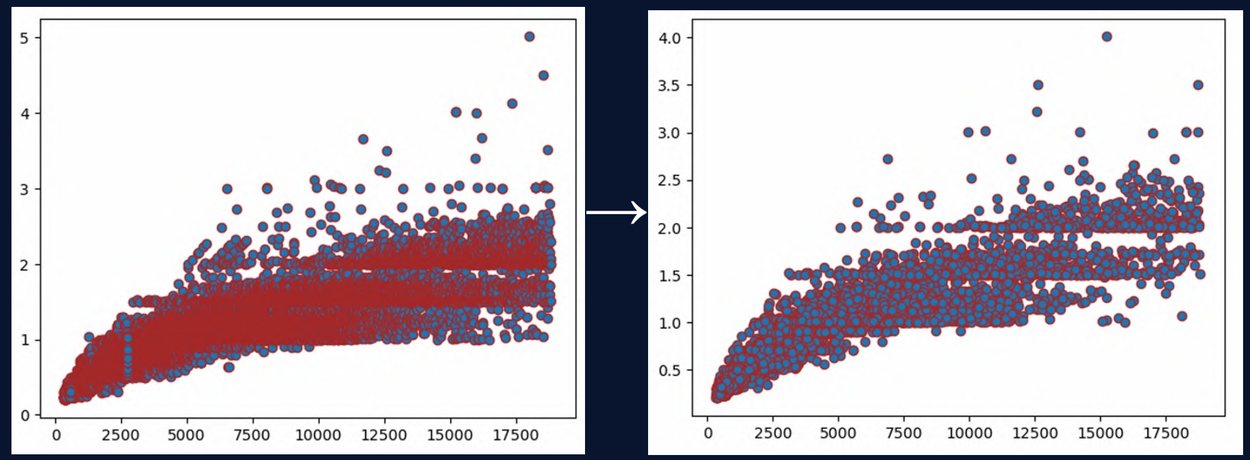


Feature Engineering

Sampling of Dataset

The original dataset of Diamonds consists of 53,940 tuples or rows along with 10 columns, resulting in a (53,940x10) sized data frame. Since data analysis of such cluttered data becomes quite difficult, a few samples, i.e., 25% of the dataset are taken to form a new dataset named 'new_diamonds' of size (13485x10).

Various data visualisation operations are performed on this new dataset considering that the new sampled dataset replicates almost the same features and distributions that the original dataset demonstrates.




```
y = new_diamonds['price']
y
```

```
new_diamonds = new_diamonds.drop('price', axis=1)
new_diamonds
```

```
new_diamonds.insert(9,'price',y,True)
new_diamonds
```

```
In [25]: X = new_diamonds.iloc[:,0:9]
         Y = new_diamonds.iloc[:,-1]
         X
```

Out[25]:

	carat	cut	color	clarity	depth	table	x	y	z
53201	0.71	2.0	2	2.0	61.5	56.0	5.76	5.69	3.52
24331	2.02	0.0	5	3.0	64.5	60.0	7.94	7.82	5.08
19051	1.06	2.0	3	7.0	62.9	56.0	6.60	6.56	4.14
51227	0.70	0.0	0	2.0	64.9	64.0	5.57	5.50	3.59
26795	2.00	2.0	1	3.0	62.2	57.0	8.11	8.09	5.04
...	...	...	...	...	...	...	...	...	...
28430	0.30	2.0	0	5.0	61.4	58.0	4.29	4.31	2.64
7536	0.70	4.0	0	6.0	62.7	54.0	5.67	5.71	3.57

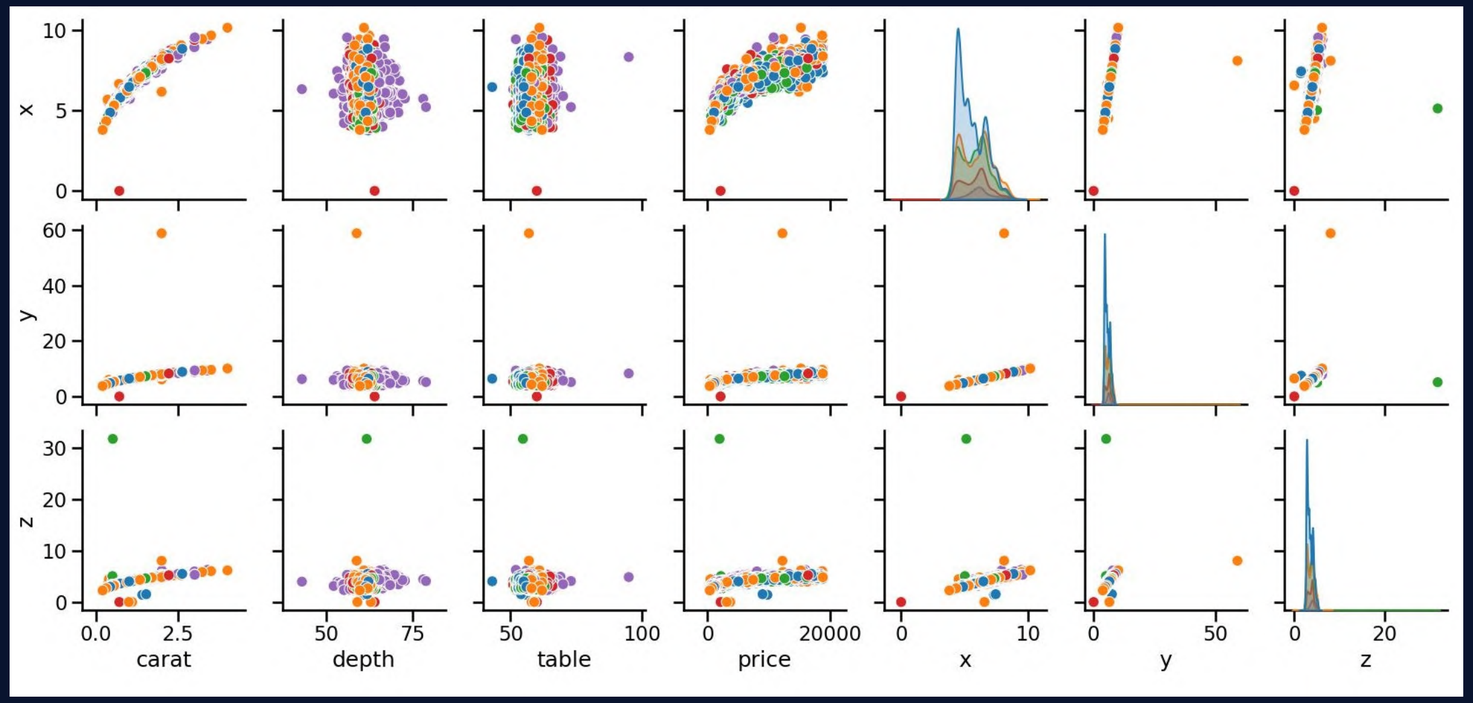
Rearrangement of Dataset

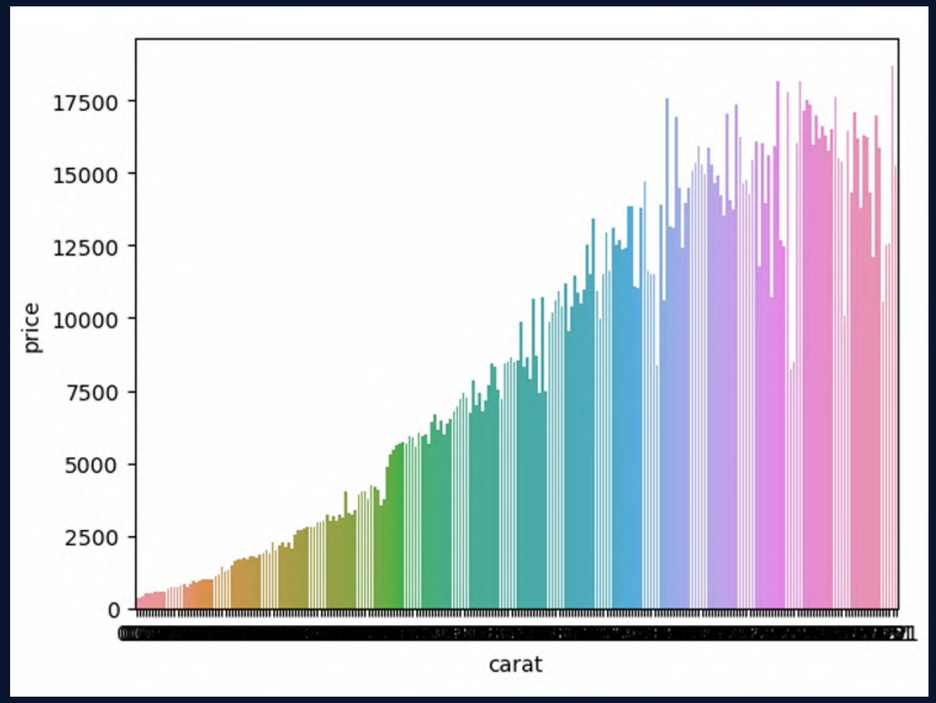
The dataset, both sampled and the original one, consists of schema of the order in such a way that it is difficult to extract independent features/attributes from dependent feature/attribute. Hence, to solve this problem, some inbuild pandas functions have been used.

Dataframe has been sliced into independent and dependent variable/s after the rearrangement performed.

Model Selection

Depending on the trends observed in the previous slides, a pair plot was created to understand the relation of every numeric data column existing with each other.





Upon observing and understanding the trends of the dataset, it can be concluded that a model based on Linear Regression may give better accuracies since most of the values showcase a linear trend. Hence, for this analysis, Linear Regression is used.

Model Training

Steps to Preprocess Data:

- ## Normalization:

Normalization in machine learning is the process of translating data into the range [0, 1] (or any other range) or simply transforming data onto the unit sphere.

- ## Standardization:

Standardization refers to setting the mean to zero and the standard deviation to one. Here, StandardScaler function has been used for stabdardization

- ## Feature Selection:

For the given dataset, all the features were selected, including the numerical and categorical data available. The categorical data was segregated into **Nominal** and **Ordinal Data** categories.

```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2
3 print("X_train.shape = ", X_train.shape)
4 print("Y_train.shape = ", Y_train.shape)
5 print("X_test.shape = ", X_test.shape)
6 print("Y_test.shape = ", Y_test.shape)
```

```
X_train.shape =  (10788, 9)
Y_train.shape =  (10788,)
X_test.shape =  (2697, 9)
Y_test.shape =  (2697,)
```

```
1 from sklearn.preprocessing import StandardScaler
2 sc = StandardScaler()
3 X_train_fit = sc.fit_transform(X_train)
4 X_test_fit = sc.transform(X_test)
```

	carat	cut	color	clarity	depth	table	x	y	z	price
0	0.23	2.0	1.0	1.0	61.5	55.0	3.95	3.98	2.43	326
1	0.21	3.0	1.0	1.0	59.8	61.0	3.89	3.84	2.31	326
2	0.23	1.0	1.0	1.0	56.9	65.0	4.05	4.07	2.31	327
3	0.29	3.0	5.0	5.0	62.4	58.0	4.20	4.23	2.63	334
4	0.31	1.0	6.0	6.0	63.3	58.0	4.34	4.35	2.75	335
...	...	...	...	...	...	...	...	...	...	...
53935	0.72	2.0	0.0	0.0	60.8	57.0	5.75	5.76	3.50	2757
53936	0.72	1.0	0.0	0.0	63.1	55.0	5.69	5.75	3.61	2757
53937	0.70	4.0	0.0	0.0	62.8	60.0	5.66	5.68	3.56	2757
53938	0.86	3.0	4.0	4.0	61.0	58.0	6.15	6.12	3.74	2757
53939	0.75	2.0	0.0	0.0	62.2	55.0	5.83	5.87	3.64	2757

53940 rows × 10 columns

Dealing with Categorical Data:

There are 2 categories of data in categorical data. These categories are:

- **Ordinal Data** - Data corresponding to an order/ranking.
- **Nominal Data** - Data which does not follow any given order/ranking

To use this data in Linear Regression predictions, we encode these **columns** or **Series** in such a way that even categorical variables can be represented in the form of numerical ranking or classes.

To encode for Ordinal Columns/Values, the **OrdinalEncoder function** has been used.

For encoding the nominal Columns/values, in this scenario, **LabelEncoder** has been used. However, it is advised to use **OneHotEncoder** for nominal values in the dataset.

```
1 from sklearn.preprocessing import OrdinalEncoder
2 encoder = OrdinalEncoder()
3 encoder.fit(np.asarray(new_diamonds['cut']).reshape(-1,1))
4 new_diamonds['cut'] = encoder.transform(np.asarray(new_diamonds['cut']
```

```
1 encoder_clarity = OrdinalEncoder()
2 encoder_clarity.fit(np.asarray(new_diamonds['clarity']).reshape(-1,1))
3 new_diamonds['clarity'] = encoder_clarity.transform(np.asarray(new_diam
```

```
1 from sklearn.preprocessing import LabelEncoder
2 encoder = LabelEncoder()
3 new_diamonds['color']=encoder.fit_transform(new_diamonds['color'])
4 new_diamonds
```

	carat	cut	color	clarity	depth	table	x	y	z	price
15164	1.31	3.0	4	5.0	59.6	58.0	7.14	7.08	4.24	6095
45273	0.54	4.0	2	5.0	59.4	57.0	5.29	5.35	3.16	1662
28326	0.33	3.0	3	4.0	61.9	58.0	4.43	4.46	2.75	666
16964	1.10	4.0	1	5.0	61.9	55.0	6.61	6.67	4.11	6776
34871	0.30	4.0	3	7.0	63.2	57.0	4.28	4.23	2.69	878
...	...	...	...	...	...	...	...	...	...	...
4412	1.01	4.0	5	2.0	58.8	58.0	6.52	6.57	3.85	3610
8318	1.00	4.0	1	3.0	63.3	55.0	6.29	6.25	3.97	4390
26588	2.40	3.0	6	2.0	59.7	58.0	8.75	8.71	5.21	16304
18211	1.00	3.0	1	4.0	58.7	62.0	6.60	6.55	3.86	7392
7146	0.93	2.0	1	2.0	62.0	56.0	6.25	6.29	3.89	4179

13485 rows × 10 columns

Results and Insights

We created 2 Linear Regression models, one for the sampled data and another for the original dataset. The resultant accuracies obtained are as follows:

On Sampled Dataset

88.752% On Training dataset

89.22% On Testing dataset

On Original Dataset

88.401% On Training dataset

88.740% On Testing dataset

Suggestions for improvements

- One way to improve the accuracy of the model is to use a new model which can be created using TensorFlow.
- Artificial Neural Networks(ANN) can be used to solve the errors existing in Linear Regression predictions and also help us increase the accuracy of the model without the model overfitting or underfitting the dataset.
- Multiple updated datasets can be referred to get the latest insights on the current ongoing trends in the market. This can be used to get accurate predictions for prices concerning the current market trends.
- Another improvement that is possible is to convolve various updated datasets from various sources and try to remove the redundancies, if any, from the dataset to increase the accuracy of the model.

Contributions

- **Karthik Sarma Dhulipati**
  Creation of Model and governance of Model accuracies with respect to sampled and original datasets
- **Saunak Saha**
  Exploratory Data Analysis and Dataset trend recognition
- **Tanisha Sarkar**
  DataSet Aquisition and Schema Analysis
- **Rishav Pandey**
  Model Improvement suggestions and analysis for better accuracy
- **Ayush Yadav**
  Methods for Standardizing and Normalizing the dataset for model creation

Thank You

-Karthik Sarma Dhulipati (2105800)
-Saunak Saha (2105660)
-Tanisha Sarkar (2105762)
-Rishav Pandey (2105733)
-Ayush Yadav (2105873)

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)