# PREDICT CAR PRICE USING NATURAL LANGUAGE PROCESSING

Group 20:
Team Members: Vishnu Sudireddy

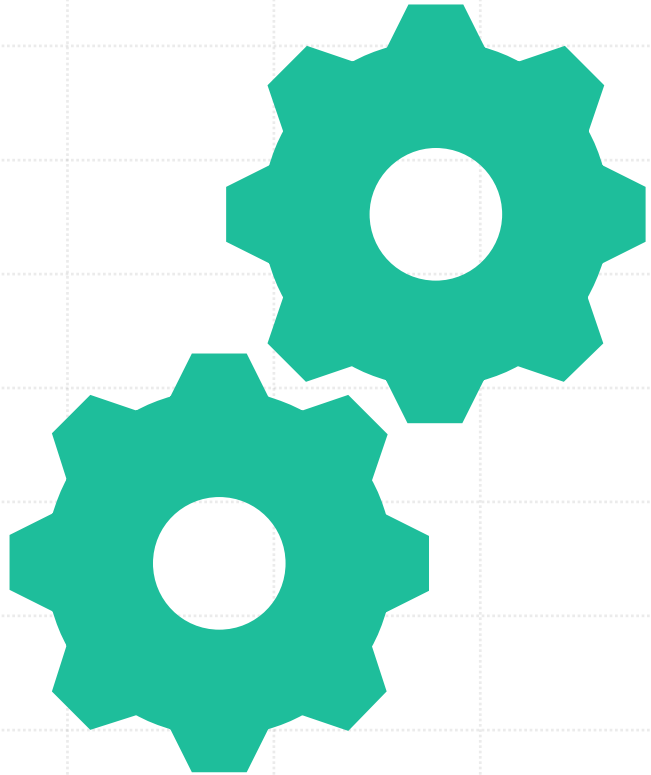
# Introduction

- Now a days cars has become an integral part of life.
- At least 60% of the people have their personal car.
- Car has life and they needs to be changed after certain period of time.

# Problem Statement and Hypothesis

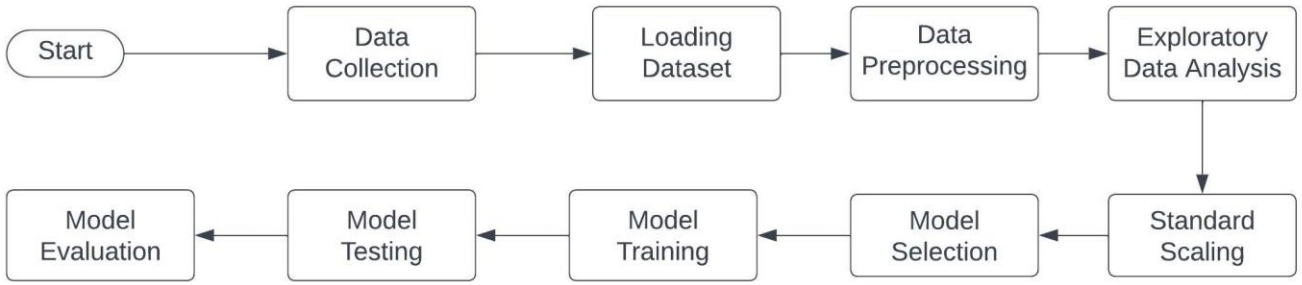
- People sell old cars on purchase of new car
- Filling forms and checking the price with the agents to sell the car is time taking process.
- Using Natural language processing along with Machine learning can be handy in predicting the price of the car.

# Methodology

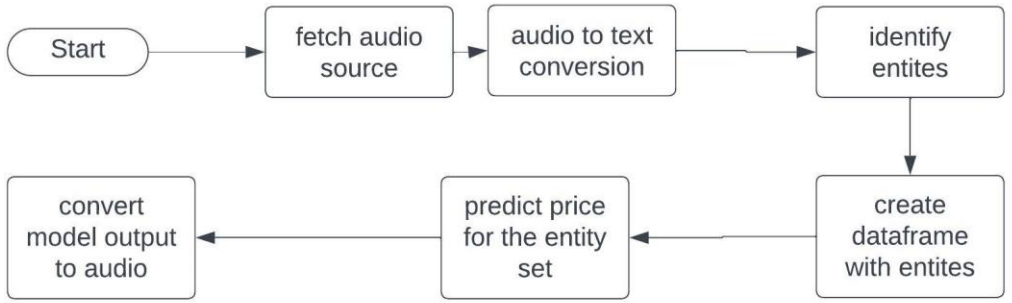


# Workflow Diagram

Level 1: ML



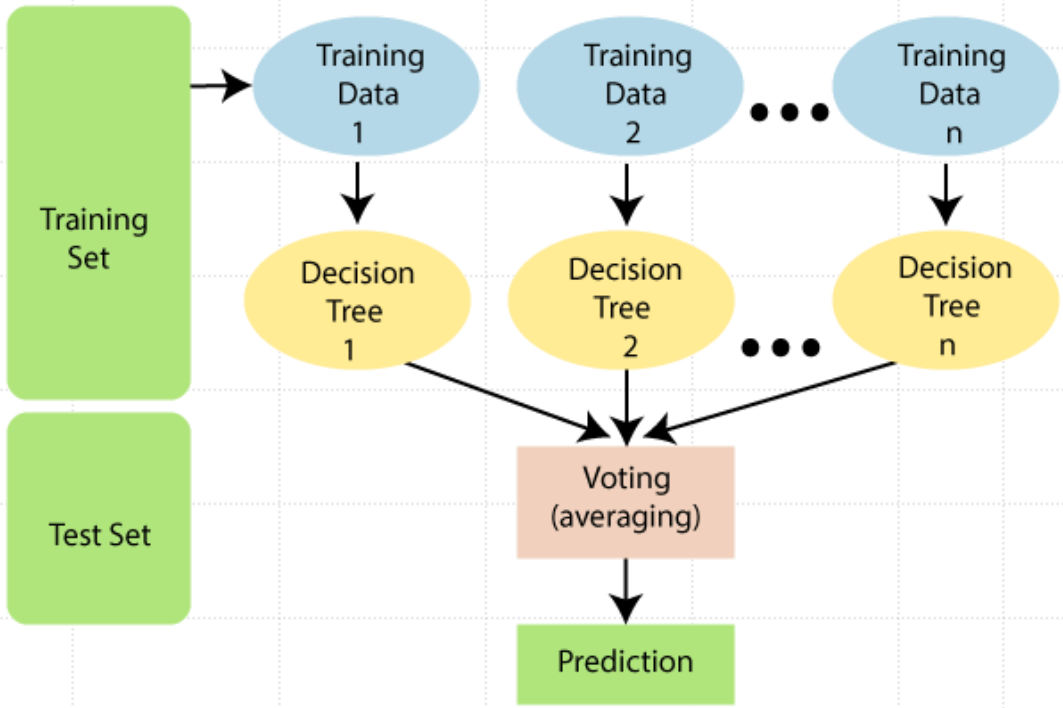

Level 2: NLP



# Architecture

## Random Forest

- The Random Forest Regressor is an ensemble machine learning algorithm that belongs to the decision tree family
- the architecture of the Random Forest Regressor leverages the power of multiple decision trees.
- Each tree is trained on different subsets of data, to create a robust and accurate predictive model.
- The randomness injected during both data and feature selection contributes to the model's stability and generalization ability.



# Dataset

- The dataset has a 16 columns and 11,914 rows
- The dataset has the information about the cars make, model, year, mileage, fuel type, transmission type, HP, size and style etc.
- The data in the dataset  has the prices of the car of different years.
- The dataset is taken from kaggle

# Loading Dataset

+ Code + Text

```
[ ]  # reading data
     cars_data = pd.read_csv('/content/sample_data/data.csv')
```

```
[ ]  #visualizing data
     cars_data.head()
```

| | Make | Model | Year | Engine Fuel Type | Engine HP | Engine Cylinders | Transmission Type | Driven_Wheels | Number of Doors | Market Category | Vehicle Size | Vehicle Style | highway MPG | city mpg | Popularity | MSRP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | BMW | 1 Series M | 2011 | premium unleaded (required) | 335.0 | 6.0 | MANUAL | rear wheel drive | 2.0 | Factory Tuner,Luxury,High-Performance | Compact | Coupe | 26 | 19 | 3916 | 46135 |
| **1** | BMW | 1 Series | 2011 | premium unleaded (required) | 300.0 | 6.0 | MANUAL | rear wheel drive | 2.0 | Luxury,Performance | Compact | Convertible | 28 | 19 | 3916 | 40650 |
| **2** | BMW | 1 Series | 2011 | premium unleaded (required) | 300.0 | 6.0 | MANUAL | rear wheel drive | 2.0 | Luxury,High-Performance | Compact | Coupe | 28 | 20 | 3916 | 36350 |
| **3** | BMW | 1 Series | 2011 | premium unleaded (required) | 230.0 | 6.0 | MANUAL | rear wheel drive | 2.0 | Luxury,Performance | Compact | Coupe | 28 | 18 | 3916 | 29450 |
| **4** | BMW | 1 Series | 2011 | premium unleaded (required) | 230.0 | 6.0 | MANUAL | rear wheel drive | 2.0 | Luxury | Compact | Convertible | 28 | 18 | 3916 | 34500 |

# Data Preprocessing

- Checked for null values
- Renamed the columns with appropriate names
- Check for duplicates and remove duplicates
- Removed the market column as it has many null values, and it may effect the model performance
- Filling the null values with appropriate possible data
- Captured the unique values of each column as they help in identifying the make, model and other details of car in the users input.
- Check for outliers and remove outliers

# Exploratory Data Analysis

- Features vs count of cars
- Horse power analysis
- Car Mileage analysis
- Car popularity analysis
- Car Price analysis
- car price vs cylinders vs year analysis
- car price vs cylinders vs HP analysis
- car price vs cylinders vs city mileage analysis

# Libraries Used

- Numpy
- Pandas
- Seaborn
- Plotly
- standardScalar
- Random forest regressor
- Speech Recognition
- Google text to Speech
- NLTK
- SentimentIntensityAnalyzer

# Implementation

- Selected features
- One-hot encoding
- Test train data split
- StandardScaling of features

```
[ ]   cat_features = ['make', 'model', 'fuel_type', 'transmission', 'drive', 'size', 'style']
      cars_data = pd.get_dummies(cars_data, columns = cat_features)
```

```
[ ]   X = cars_data.drop('price', axis = 1)
      y = cars_data['price']

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
```

```
[ ]   sc = StandardScaler()
      X_train = sc.fit_transform(X_train)
      X_test = sc.transform(X_test)
```

# Implementation (continued..)

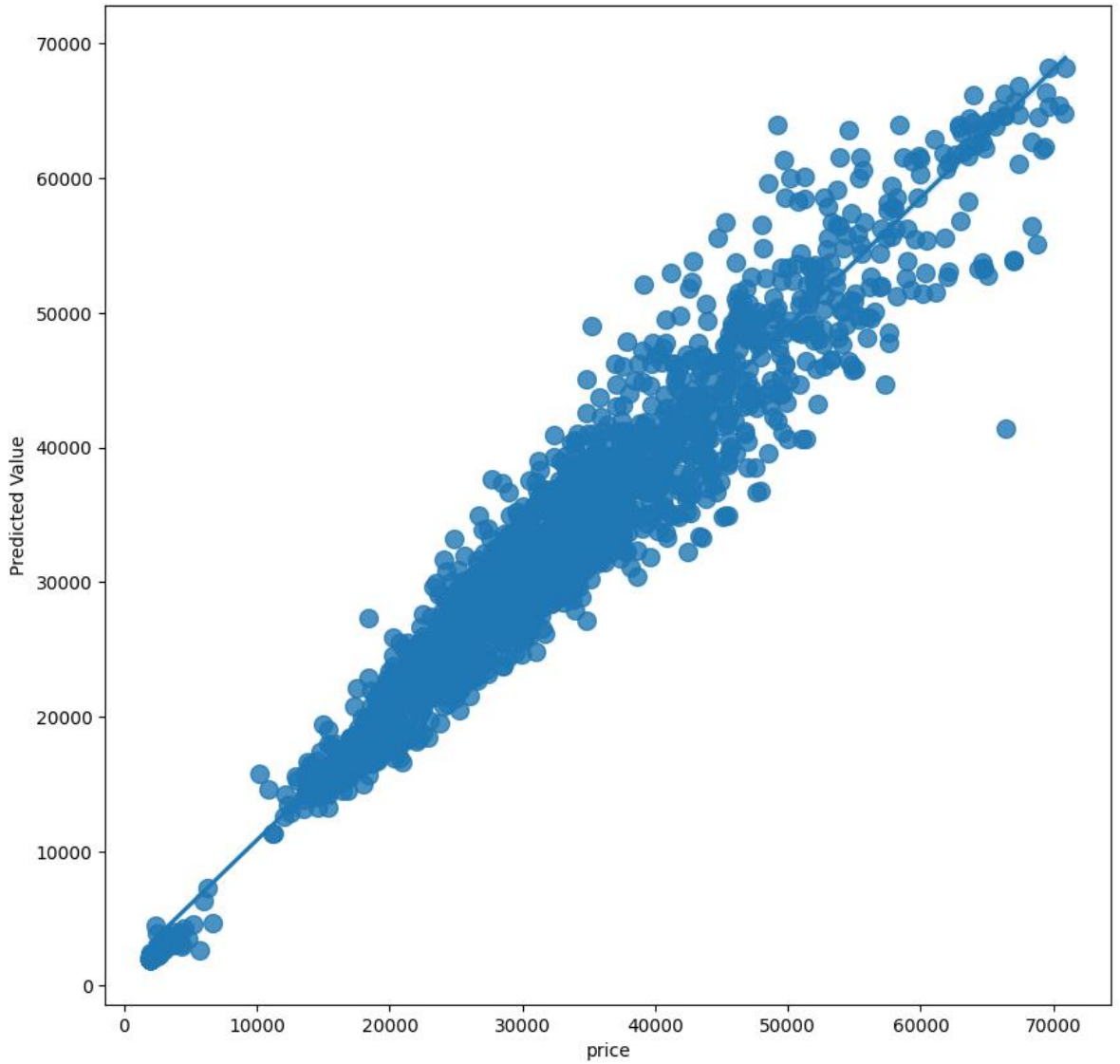
```
rfr = RandomForestRegressor(n_estimators = 40)
rfr_algo = make_pipeline(rfr)

rfr_algo.fit(X_train, y_train)
rfr_pred = rfr_algo.predict(X_test)

print('R2 Score is : ', r2_score(y_test, rfr_pred))
print('Mean squared error is : ', math.sqrt(mean_squared_error(y_test, rfr_pred)))
```

```
R2 Score is :  0.9536431963599306
Mean squared error is :  3339.190959820919
```

# Predicted Price vs Actual Price



# Audio to text conversion

- Used Speech_recognition library
- Created an audio file and provided as input
- Converted audio to text using recognizer_google method from speech_recognition library

```
import speech_recognition as sr

audio_file_path = "/content/sample_data/negative_sell.wav"
recognizer = sr.Recognizer()

try:
    with sr.AudioFile(audio_file_path) as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.record(source)
        audio_text = recognizer.recognize_google(audio)
        print("Transcription:", audio_text)
except sr.UnknownValueError:
    print("Could not understand audio")
except Exception as e:
    print(f"An error occurred: {e}")
```

```
Transcription: choose with my car due to various problems and I want to sell it can you predict the price of 2011 compact Coupe BMW 1 Series M model which takes premium unleaded as fuel it is a ma
```

# Sentiment analysis

Performed sentiment analysis on users input data using sentiment intensity analyzer

```python
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

nltk.download('vader_lexicon')  # Download the VADER lexicon

def analyze_sentiment(sentence):
    sid = SentimentIntensityAnalyzer()
    sentiment_scores = sid.polarity_scores(sentence)

    if sentiment_scores['compound'] >= 0.05:
        return "Positive"
    elif sentiment_scores['compound'] <= -0.05:
        return "Negative"
    else:
        return "Neutral"

# Example usage:
sentiment = analyze_sentiment(sentence)
print(sentence)

print(f"Sentiment: {sentiment}")
```

```
choose with my car due to various problems and i want to sell it can you predict the price of 2011 compact coupe bmw 1 series m model which takes premium unleaded as fuel it is a manual rear wheel
Sentiment: Negative
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

# Entity Recognition

- Identified the vehicles information in the users input and added it to a data structure.

```
target_vehicle = {
    'make': '',
    'model': '',
    'fuel_type': '',
    'transmission':'',
    'drive': '',
    'size': '',
    'style': ''
}
for item in information['make']:
  if item.lower() in sentence:
    target_vehicle['make'] = item
    break

for item in information['model']:
  if item.lower() in sentence:
    target_vehicle['model'] = item
    break
```

```
for item in information['fuelType']:
  if item.lower() in sentence:
    target_vehicle['fuel_type'] = item
    break

for item in information['transmission']:
  if item.lower() in sentence:
    target_vehicle['transmission'] = item
    break

for item in information['drive']:
  if item.lower() in sentence:
    target_vehicle['drive'] = item
    break


for item in information['size']:
  if item.lower() in sentence:
    target_vehicle['size'] = item
    break

for item in information['style']:
  if item.lower() in sentence:
    target_vehicle['style'] = item
    break
```

# Predicting the Price

The data structure passed to predict_car_price method.

This method will convert dictionary datastructure to pandas dataframe  and provide dataframe as input to model.

The model will predict the price for the provided details of the car.

```
predicted_price = predict_car_price(target_vehicle, rfr_algo, cat_features, sc, X)
```

```
import pandas as pd

def predict_car_price(user_input, rfr_algo, cat_features, sc, X):
    # Create a DataFrame from the user input
    user_input_df = pd.DataFrame([user_input])

    # One-hot encode categorical features
    user_input_df = pd.get_dummies(user_input_df, columns=cat_features)

    # Make sure the user input DataFrame has the same columns as the training data
    user_input_df = user_input_df.reindex(columns=X.columns, fill_value=0)

    # Standardize the input data using the same scaler
    user_input_scaled = sc.transform(user_input_df)

    # Use the trained model to make predictions
    predicted_price = rfr_algo.predict(user_input_scaled)

    return predicted_price[0]
```

# Result

- The predicted output is converted to audio using googles text to speech convertion library

```
from gtts import gTTS
import os

def text_to_speech(text, language='en', filename='output.mp3'):
    tts = gTTS(text=text, lang=language, slow=False)
    tts.save(filename)
    os.system(f"start {filename}")  # This command opens the file with the default audio player

# Example usage:
text = f'the predicted cost of the car is ${predicted_price}'
text_to_speech(text)
```

# Result

- The result obtained with this model has achieved r-squared score as 0.95  and the mean square error as 3391.

```
[ ]    rfr = RandomForestRegressor(n_estimators = 40)
       rfr_algo = make_pipeline(rfr)

       rfr_algo.fit(X_train, y_train)
       rfr_pred = rfr_algo.predict(X_test)

       print('R2 Score is : ', r2_score(y_test, rfr_pred))
       print('Mean squared error is : ', math.sqrt(mean_squared_error(y_test, rfr_pred)))
```

```
R2 Score is :  0.9536431963599306
Mean squared error is :  3339.190959820919
```

# Project Management

- In this Project, I am the only team member and I have taken the responsibility of all the tasks starting from dataset identification to  data cleaning, data preprocessing, model selection and building and generating the results.

# References/Bibliography


- [1] https://www.kaggle.com/datasets/CooperUnion/cardataset/data
- [2] https://www.kaggle.com/datasets/ravishah1/carvana-predict-car-prices
- [3] https://cloud.google.com/text-to-speech
- [4] https://pypi.org/project/SpeechRecognition/
- [5] https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

# Thankyou