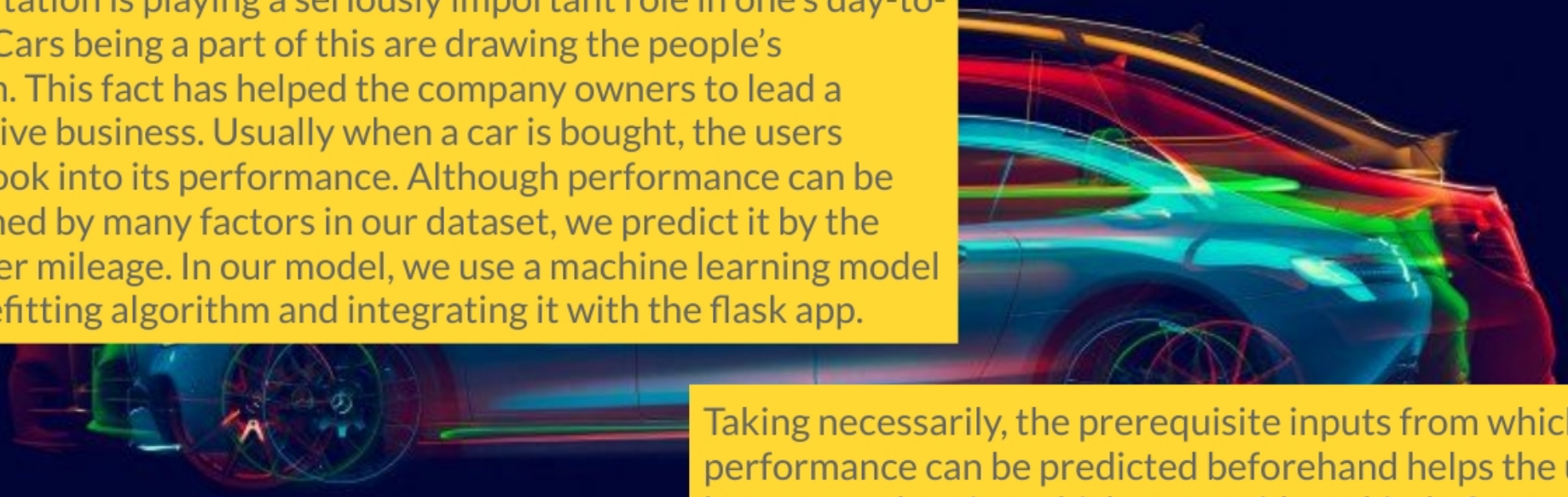
CAR PERFORMANCE PREDICTOR

TEAM - 31

ABOUT THE PROJECT

Transportation is playing a seriously important role in one's day-to-day life. Cars being a part of this are drawing the people's attention. This fact has helped the company owners to lead a progressive business. Usually when a car is bought, the users usually look into its performance. Although performance can be determined by many factors in our dataset, we predict it by the parameter mileage. In our model, we use a machine learning model with a befitting algorithm and integrating it with the flask app.



Taking necessarily, the prerequisite inputs from which performance can be predicted beforehand helps the user to learn a good option which can avoid any kind of interruptions in their journey later on and so. This performance prediction is done with the concept of machine learning that has highest accuracy infers the finest algorithm.

TARGET PROBLEM

The problem

Selecting the best car to buy (with highest performance) is a complex task for laymen.

The solution

Elegant and simple application that asks the users to enter the car's data and get it's performance as the result.

HOW WE SOLVED THE PROBLEM

1. DATA PROCESSING

Download car dataset → Drop '?' values in columns 'horsepower' → split data into dependent and independent

Independent: Input values

Dependant: output mpg/ performance.

2. MACHINE LEARNING

split data into train and test set → train and test the model

Random forest regression

3. WEB DEVELOPMENT

pickle file

HTML code

python 'app.py.

4. APP HOSTING

CAR PERFORMANCE PREDICITION APPLICATION

PROJECT FLOWCHART.

1. DATA PROCESSING : USING PYTHON LIBRARIES VIA JUPYTER NOTEBOOK


```
In [1]: import pandas as pd
        import numpy as np
```

```
In [2]: datas = pd.read_csv(r"C:\Users\Sravanya Mogalluru\Desktop\Python Project\car performance.csv")
```

```
In [3]: type(datas)
Out[3]: pandas.core.frame.DataFrame
```

```
In [4]: datas.isnull().any()
Out[4]: mpg             False
        cylinders       False
        displacement    False
        horsepower      False
        weight          False
        acceleration    False
        model year      False
        origin          False
        car name        False
        dtype: bool
```

```
In [5]: datas.head(399)
```

Out[5]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

```
In [6]: datas['car name'].unique()
Out[6]: array(['chevrolet chevelle malibu', 'buick skylark 320',
               'plymouth satellite', 'amc rebel sst', 'ford torino',
               'ford galaxie 500', 'chevrolet impala', 'plymouth fury iii',
```

```
In [7]: x = datas.iloc[:,1:8].values

In [8]: y = datas.iloc[:,0].values
```

```
In [9]: 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)

In [10]: x.shape

Out[10]: (398, 7)

In [11]: x_train

Out[11]: array([[  8. , 318. , 150. , ...,  13.5,  72. ,   1. ],
                [  4. ,  97. ,  60. , ...,  19. ,  71. ,   2. ],
                [  4. ,  97. ,  78. , ...,  15.8,  80. ,   2. ],
                ...,
                [  4. ,  68. ,  49. , ...,  19.5,  73. ,   2. ],
                [  6. , 250. , 100. , ...,  15. ,  71. ,   1. ],
                [  4. ,  90. ,  71. , ...,  16.5,  75. ,   2. ]])
```

2. MACHINE LEARNING : USING PYTHON LIBRARIES VIA JUPYTER NOTEBOOK
(M.L. ALGORITHM : RANDOM FOREST REGRESSION)

In [9]:
```
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)
```

In [10]:
```
x.shape
```

Out[10]:
```
(398, 7)
```

In [11]:
```
x_train
```

Out[11]:
```
array([[  8. , 318. , 150. , ...,  13.5,  72. ,   1. ],
       [  4. ,  97. ,  60. , ...,  19. ,  71. ,   2. ],
       [  4. ,  97. ,  78. , ...,  15.8,  80. ,   2. ],
       ...,
       [  4. ,  68. ,  49. , ...,  19.5,  73. ,   2. ],
       [  6. , 250. , 100. , ...,  15. ,  71. ,   1. ],
       [  4. ,  90. ,  71. , ...,  16.5,  75. ,   2. ]])
```

In [12]:
```
from sklearn.preprocessing import StandardScaler
sd = StandardScaler()
x_train = sd.fit_transform(x_train)
x_test = sd.fit_transform(x_test)
```

In [13]:
```
x_train
```

Out[13]:
```
array([[ 1.49526939,  1.22961301,  1.24320117, ..., -0.79520768,
        -1.13752513, -0.73301171],
       [-0.85285735, -0.92367663, -1.16299465, ...,  1.24411524,
        -1.41177304,  0.5068698 ],
       [-0.85285735, -0.92367663, -0.68175548, ...,  0.05760009,
         1.05645814,  0.5068698 ],
       ...,
       [-0.85285735, -1.206235  , -1.45708525, ...,  1.42950823,
        -0.86327722,  0.5068698 ],
       [ 0.32120602,  0.56706235, -0.09357428, ..., -0.2390287 ,
        -1.41177304, -0.73301171],
       [-0.85285735, -0.99188037, -0.86890405, ...,  0.31715028,
        -0.31478141,  0.5068698 ]])
```

In [14]:
```
from sklearn.ensemble import RandomForestRegressor
d = RandomForestRegressor (n_estimators=30,random_state = 0)
d.fit(x_train,y_train)
```

Out[14]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=30, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

In [15]:
```
r = d.predict(x_test)
```

```
In [16]: r
```

```
Out[16]: array([14.38333333, 24.63      , 13.96666667, 20.52      , 18.53333333,
                30.18333333, 34.37      , 21.14333333, 15.72333333, 25.45666667,
                36.4       , 36.03333333, 19.67666667, 27.32333333, 16.62333333,
                32.58      , 28.31      , 27.43      , 17.42666667, 36.12333333,
                16.71333333, 23.49      , 23.30333333, 20.67333333, 33.50666667,
                26.22666667, 33.05666667, 30.21333333, 31.38      , 16.41333333,
                20.30333333, 33.28666667, 20.58      , 33.95666667, 20.91666667,
                24.34      , 19.55      , 17.39      , 34.27666667, 12.65      ,
                14.18333333, 15.23333333, 28.71666667, 32.98666667, 28.47333333,
                22.76      , 20.59333333, 16.7       , 23.39666667, 29.98333333,
                34.21      , 26.15      , 17.71666667, 27.81666667, 16.1       ,
                13.1       , 18.83333333, 26.6       , 31.21333333, 15.79      ,
                20.4       , 25.99      , 20.37666667, 21.70666667, 13.5       ,
                15.35      , 14.28333333, 19.02666667, 24.70333333, 14.58333333,
                34.83666667, 13.18333333, 22.92666667, 18.92333333, 23.46666667,
                31.97333333, 26.85      , 31.21333333, 31.79      , 14.35      ])
```

```
In [17]: from sklearn.metrics import r2_score
         accuracy = r2_score(y_test,r)
```

```
In [18]: accuracy
```

```
Out[18]: 0.888732160761456
```

```
from sklearn.svm import SVR
svm = SVR(kernel = 'linear')
svm.fit(x_train,y_train)
```

```
y_pre = svm.predict(x_test)
```

```
from sklearn.metrics import r2_score
accurac = r2_score(y_test,y_pre)
```

```
accurac
```

```
In [21]: d.predict([[1,6,17,34,11,123,1]])
```

```
Out[21]: array([17.17])
```

3. WEB DEVELOPMENT: USING FLASK FRAMEWORK IN SPYDER ENVIRONMENT (PICKLE FILES FOR M.L. INTEGRATION)

app.py

```python
from flask import Flask , render_template , request
import pickle

app = Flask(__name__)
model = pickle.load(open('regression.pkl','rb'))

@app.route('/')
def intro():
    return render_template('index.html')
@app.route('/login', methods = ["POST"])
def login():

    cyl = request.form["cyl"]
    dis = request.form["dis"]
    hp = request.form["hp"]
    w = request.form["w"]
    a = request.form["a"]
    my = request.form["my"]
    ori = request.form["ori"]
    total = [[int(cyl),int(dis),int(hp),int(w),int(a),int(my),int(ori)]]

    p = model.predict(total)
    p =p[0]
    return render_template('index.html',label = "The performance of the car is  "+ str(p))

if __name__=='__main__':
    app.run(debug = True,port = 9000)
```

app.py Flask code

HTML code

index.html

templates > index.html > html

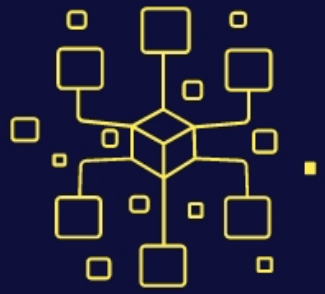
```
<html>
<style>
.background-image {
  position: fixed;
  left: 0;
  right: 0;
  z-index: 1;
  display: block;
  background-image: url('https://www.mercedes-benz.com/en/vehicles/passenger-cars/_jcr_content/root/grid_copy/grid-par/griditem_copy/image/MQ6-8-image-20200129142634/00-
  width: 12000px;
  height: 200000px;
  -webkit-filter: blur(3px);
  -moz-filter: blur(3px);
  -o-filter: blur(3px);
  -ms-filter: blur(3px);
  filter: blur(3px);
}
.content {
  position: absolute;
  top: 50%
  left: 50%;
  right: 50%;
  z-index: 2;
  width: 20%;
  content: center;
  }

  </style>
```

HTML code

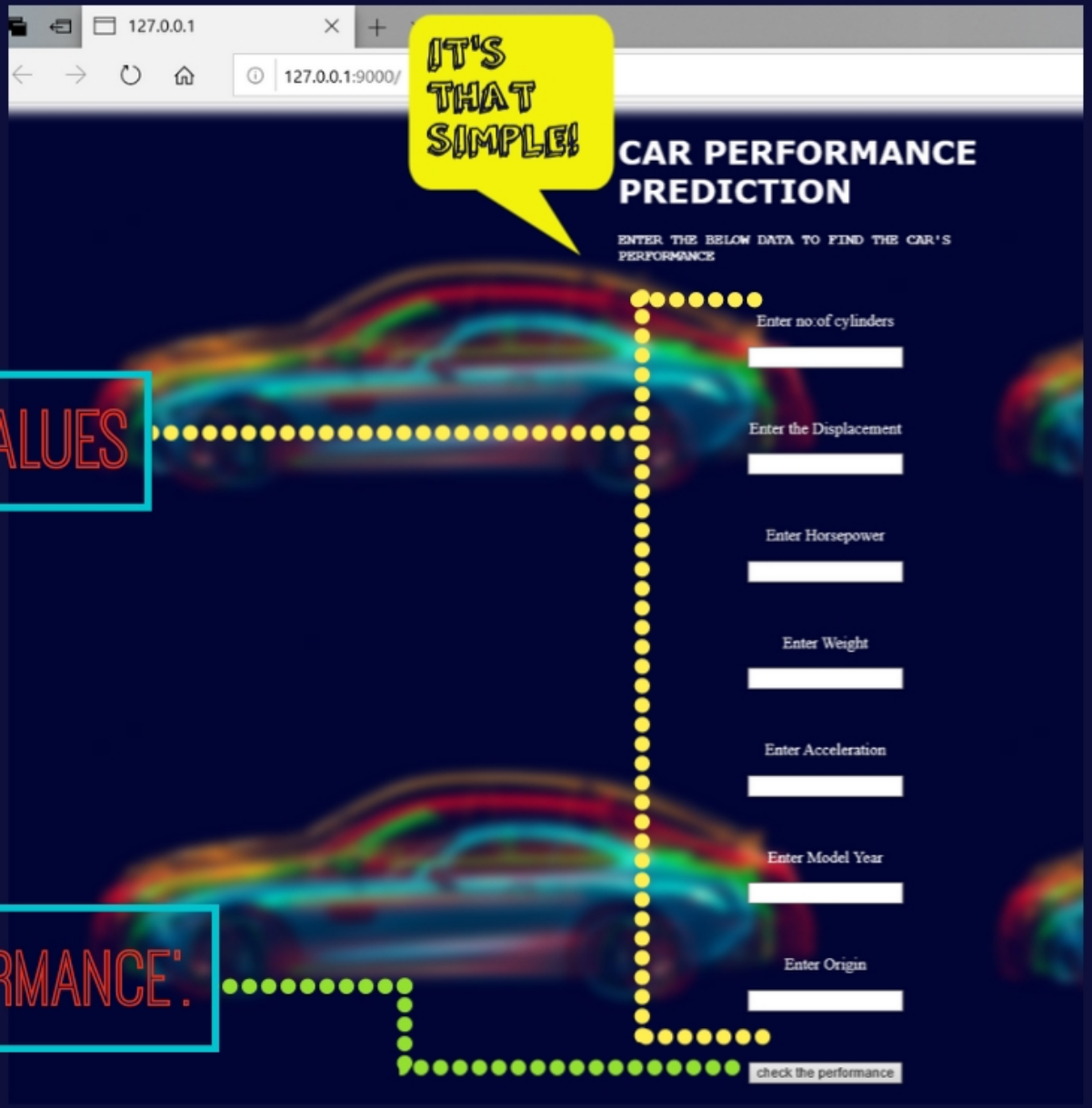
```
<body>
<form action ="http://localhost:9000/login" method="post">
<div class="background-image">  </div>
<div class="content">
<header>
<h1 style="font-family:verdana;font-size:200%;content-align:center;color:white;"> CAR PERFORMANCE PREDICTION </h1>
<h2 style="font-family:courier;font-size:90%;content-align:center;color:white;"> ENTER THE BELOW DATA TO FIND THE CAR'S PERFORMANCE </h2>
<br>
<p style="text-align:center;color:white;"> Enter no:of cylinders </p>
<p style="text-align:center;color:white;"><input type= "text" name="cyl"/></p>
<br>
<p style="text-align:center;color:white;"> Enter the Displacement</p>
<p style="text-align:center;color:white;"> <input type= "text" name="dis"/></p>
<br>
<p style="text-align:center;color:white;"> Enter Horsepower</p>
<p style="text-align:center;color:white;"> <input type= "text" name="hp"/></p>
<br>
<p style="text-align:center;color:white;"> Enter Weight</p>
<p style="text-align:center;color:white;"><input type= "text" name="w"/></p>
<br>
<p style="text-align:center;color:white;"> Enter Acceleration</p>
<p style="text-align:center;color:white;"> <input type= "text" name="a"/></p>
<br>
<p style="text-align:center;color:white;"> Enter Model Year</p>
<p style="text-align:center;color:white;"> <input type= "text" name="my"/></p>
<br>
<p style="text-align:center;color:white;"> Enter Origin</p>
<p style="text-align:center;color:white;"> <input type= "text" name="ori"/></p>
<br>
<p style="text-align:center;color:white;"> <input type="submit" value ="check the performance"/></p>
</form>
<b style="text-align:center;color:white;"> {{label}} </b>

</div>
</body>
</html>
```

4. APPLICATION INTERFACE: USING BROWSER

JUST PLUG IN THESE VALUES



AND CLICK ON 'CHECK THE PERFORMANCE'.



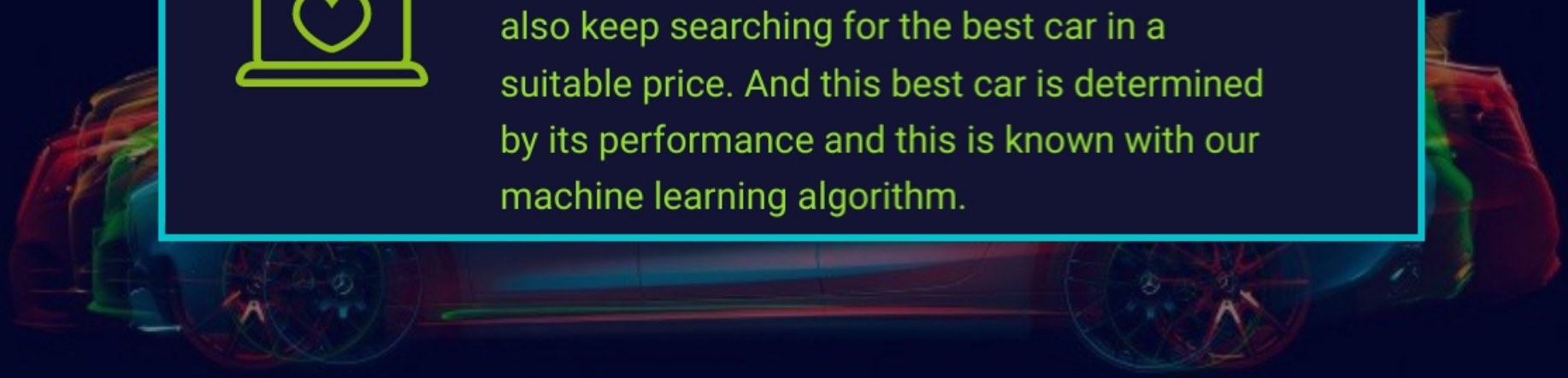
ADVANTAGES

1. Accurate predictions
2. Easy usable interface
3. Results shown are without any errors
4. It gives information about mileage as well as performance

DISADVANTAGES

1. For the prediction we need many inputs and sometimes it may lead to some uncertainty in the data or there might be any missing values which need to be taken care about.
2. Accuracy can hardly be 100 %. So, sometimes the predictions may be wrong.

CONCLUSION



With the growing population, the demand for cars has also been increasing. So the people also keep searching for the best car in a suitable price. And this best car is determined by its performance and this is known with our machine learning algorithm.

TEAM MEMBERS

PADMINI YADAV

MOGALLURU VENKATA SRAVANYA

ASHIKA KINTALI

KANNEBOYINA VAISHNAVI

PROJECT GITHUB LINK: https://github.com/SmartPracticeschool/IISPS-INT-2421-Car-Performance-Prediction


Thank You!