

---

# ISETB: Professional Master in Advanced Robotics and Artificial Intelligence

---



---

## PROJET REPORT about USED CARS PRICE PREDICTION APPLICATION

---

Leila Megdiche & Iheb Mechergui  
May 30, 2023

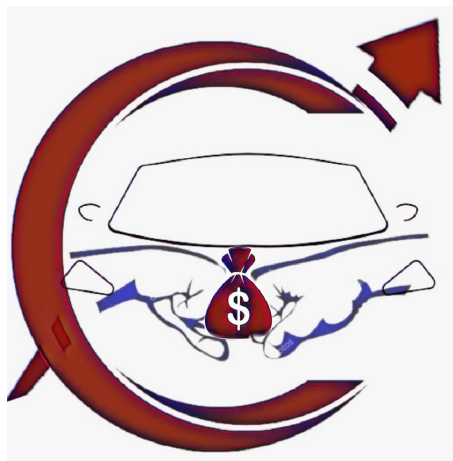


Figure 1: The Logo Of our Application

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	general introduction . . . . .	4
1.2	the subject of our Project . . . . .	5
<b>2</b>	<b>Software and platforms used:</b>	<b>7</b>
2.1	Anaconda and Jupyter notebook . . . . .	7
2.2	VS code . . . . .	7
<b>3</b>	<b>Dataset Description</b>	<b>8</b>
<b>4</b>	<b>Data Preprocessing</b>	<b>9</b>
4.1	collect of information: . . . . .	9
4.2	Handling Missing Values: . . . . .	10
4.3	Encoding Categorical Variables: . . . . .	10
4.4	remove the suffix: . . . . .	11
<b>5</b>	<b>Model Training</b>	<b>11</b>
5.1	Data Exploration: . . . . .	11
5.2	Splitting the Data: . . . . .	12
<b>6</b>	<b>Evaluation Metrics</b>	<b>13</b>
<b>7</b>	<b>Results and Discussion</b>	<b>14</b>
7.1	he result of prediction . . . . .	14
7.2	connect the code of Jupyter with Flask file: . . . . .	14
7.3	the web page: . . . . .	15
<b>8</b>	<b>Conclusion</b>	<b>15</b>
<b>9</b>	<b>Upgrades to our project:</b>	<b>16</b>
<b>10</b>	<b>References</b>	<b>18</b>

## List of Figures

1	The Logo Of our Application . . . . .	1
2	machine learning . . . . .	5
3	Jupyter LOGO . . . . .	7
4	latex LOGO . . . . .	7
5	my dataset . . . . .	8
6	upload data . . . . .	9
7	information about our data . . . . .	10
8	cleaning the data . . . . .	10
9	changing categorical features to numerical . . . . .	11
10	remove the suffix . . . . .	11
11	distribution of age . . . . .	12
12	distribution of selling-price vs other . . . . .	12
13	Split the data . . . . .	13
14	Models summarization . . . . .	13
15	the price prediction . . . . .	14
16	the flask code . . . . .	14
17	the link to the navigator . . . . .	15
18	the first interface of app . . . . .	15
19	the model on azure . . . . .	17
20	the formula of application on azure . . . . .	17
21	the price prediction result . . . . .	18

## Abstract

Machine learning (ML) is a branch of artificial intelligence (AI) that enables computers to “self-learn” from training data and improve over time, without being explicitly programmed. Machine learning algorithms are able to detect patterns in data and learn from them, in order to make their own predictions. In short, machine learning algorithms and models learn through experience.

# 1 Introduction

## 1.1 general introduction

Machine learning is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn and make predictions or decisions without being explicitly programmed. It involves the extraction of meaningful patterns from data, allowing machines to learn from examples and experiences.

In the context of prediction, machine learning algorithms can analyze historical data and identify patterns or relationships that can be used to predict future outcomes. This has various applications, including predicting prices, stock market trends, customer behavior, disease diagnosis, and more.

One of the major benefits of machine learning in predicting prices is its ability to handle complex and large datasets. Machine learning algorithms can automatically analyze vast amounts of data, identify relevant features, and build predictive models. This allows for more accurate predictions and insights compared to traditional statistical methods.

Machine learning also has the advantage of adaptability and continuous improvement. As new data becomes available, machine learning models can be retrained and updated to incorporate the latest information. This flexibility enables models to adapt to changing conditions and improve their predictions over time.

Furthermore, machine learning can uncover hidden patterns and relationships in data that may not be apparent to humans. It can capture complex nonlinear interactions and dependencies, leading to more accurate and nuanced predictions. This can be particularly beneficial in scenarios where traditional analytical approaches may fall short.

Overall, machine learning offers powerful tools and techniques for prediction tasks, such as price prediction. By leveraging large datasets, sophisticated algorithms, and continuous learning, machine learning can provide valuable insights and improve decision-making

in various domains.

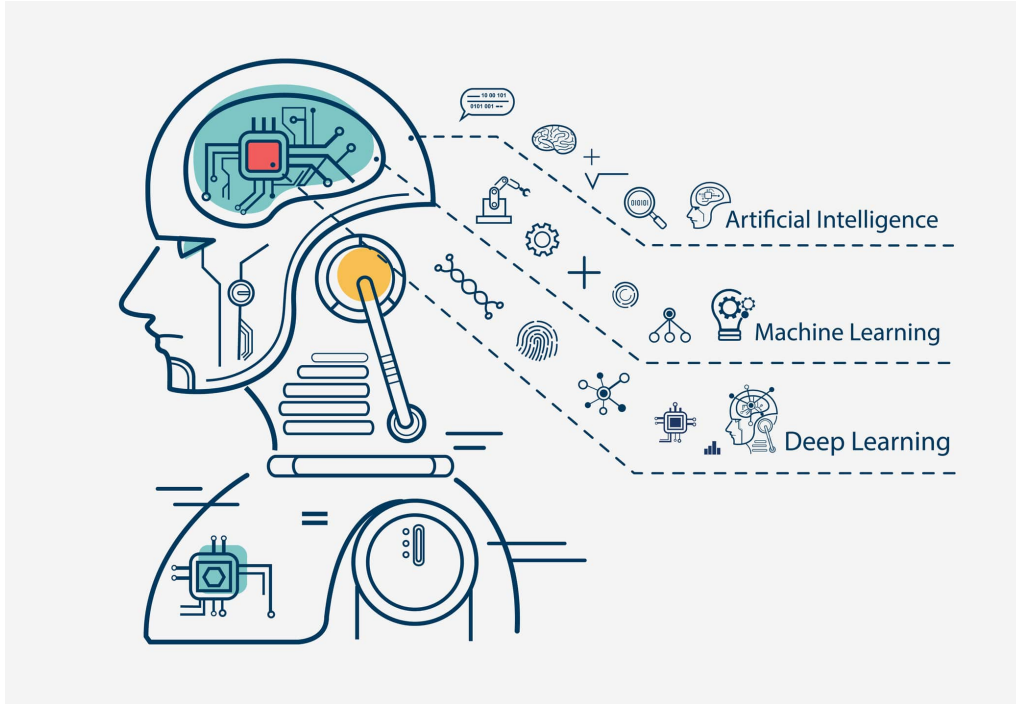


Figure 2: machine learning

## 1.2 the subject of our Project

The used car price prediction application is a project aimed at providing accurate and reliable predictions of car prices based on various factors. The purpose of this application is to assist potential car buyers and sellers in making informed decisions by estimating the fair market value of a used car. By leveraging machine learning algorithms and historical car data, the application aims to offer valuable insights into pricing trends and help users negotiate better deals.

### **Objectives:**

#### ❖ Accurate Price Estimation:

The primary objective of the application is to provide precise estimates of used car prices. By analyzing multiple factors such as car age, mileage, brand, model, and other relevant features, the application aims to generate reliable predictions that closely align with the current market value.

#### ❖ User-Friendly Interface:

Another objective is to create a user-friendly interface that allows users to input the necessary information about a car and obtain the predicted price conveniently. The application should be intuitive and accessible to users with varying levels of technical knowledge.

**Significance:**

Accurately predicting car prices is of significant importance for both potential car buyers and sellers. Here are some key reasons why this application holds significance:

**✳ Informed Decision Making:**

For potential car buyers, knowing the fair market value of a used car can help in making informed purchasing decisions. It provides a benchmark against which the listed price can be evaluated, ensuring that buyers pay a fair price and avoid overpaying.

**✳ Effective Pricing Strategy:**

For sellers, accurately predicting car prices is crucial in setting the right selling price. It ensures that the car is priced competitively, attracting potential buyers while maximizing the seller's profit. It helps sellers avoid underpricing or overpricing their vehicles, leading to a more efficient selling process.

**✳ Transparency and Trust:**

Price transparency builds trust between buyers and sellers. When both parties have access to reliable price predictions, it reduces information asymmetry and fosters trust in the used car market. This transparency can lead to more efficient transactions and a healthier marketplace.

**✳ Time and Cost Savings:**

By utilizing a car price prediction application, potential buyers and sellers can save time and effort that would otherwise be spent on extensive market research. The application automates the process and provides quick and accurate results, enabling users to make faster decisions and potentially save money.

**Conclusion:**

The used car price prediction application uses machine learning techniques to provide accurate price estimates for used cars. By empowering potential buyers and sellers with reliable pricing information, the application contributes to informed decision-making, transparency, and trust in the used car market. With the ability to estimate prices accurately, users can make more confident buying and selling decisions, leading to a more efficient and fair marketplace for used cars.

## 2 Software and platforms used:

### 2.1 Anaconda and Jupyter notebook

o write the code in python we chose Anaconda and Jupiter notebook  
Anaconda is a distribution of Python and R for scientific computing and data science. It comes with a package manager called Conda, which makes it easy to install and manage libraries and dependencies. Anaconda also includes several useful tools such as Jupyter Notebook and Spyder for data exploration and development.

Jupyter Notebook is an open-source web-based interactive development environment (IDE) that allows users to create and share documents that contain live code, equations, visualizations, and narrative text. It supports over 40 programming languages including Python, R, and Julia. Jupyter Notebook is commonly used in data science, machine learning, and scientific computing for data cleaning, transformation, visualization, and analysis.



Figure 3: Jupyter LOGO

### 2.2 VS code

we use VScode to code the  $\text{\LaTeX}$ file to create the report and the perposal.



Figure 4: latex LOGO

### 3 Dataset Description

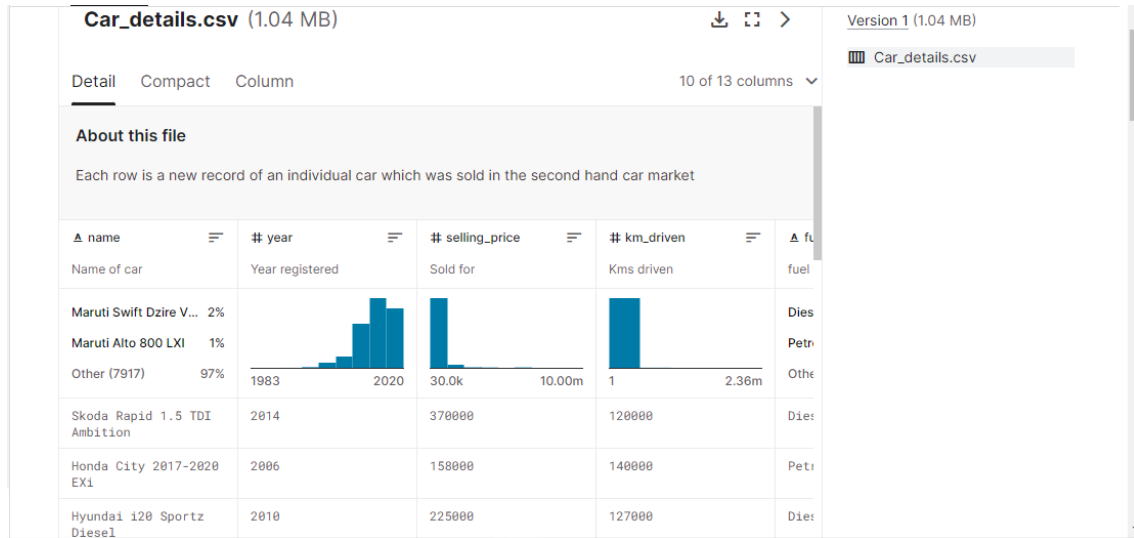


Figure 5: my dataset

The dataset contains information about used cars, including various features (independent variables) and the target variable (car price).

Here are the details of the dataset:

**1-Size of the Dataset:** The dataset consists of 8128 records (rows) and 13 columns. Each record represents a specific used car listing.

**2-Features (Independent Variables):** The dataset includes the following features:

- ◆ name: The name of the car model (e.g., "Maruti Swift", "Honda City").
- ◆ year: The manufacturing year of the car (e.g., 2015, 2017).
- ◆ selling\_price: The selling price of the car (in lakhs).
- ◆ km\_driven: The distance covered by the car in kilometers.
- ◆ fuel\_type: The type of fuel used by the car (e.g., Petrol, Diesel, CNG).
- ◆ seller\_type: The type of seller (e.g., Dealer, Individual).
- ◆ Transmission: The type of transmission system (e.g., Manual, Automatic).
- ◆ owner: The number of previous owners of the car (0, 1, 3+).
- ◆ mileage: The car's fuel efficiency in kilometers per liter.
- ◆ Engine: The engine displacement of the car in cubic centimeters (cc).
- ◆ max\_power: The maximum power of the car in bhp (brake horsepower).



- ◆ seats: The number of seats in the car.
  - ◆ torque: refers to a measure of rotational force or twisting force that an engine can generate.
- 3-Target Variable (Car Price):** The target variable is the selling\_price, which represents the selling price of the used car.

```
[2]: {'divide': 'warn', 'over': 'warn', 'under': 'ignore', 'invalid': 'warn'}

[3]: import warnings
warnings.filterwarnings('ignore')

[4]: car_data = pd.read_csv("C:/Users/megdiche leila/Desktop/MLproject/Car_details.csv")
car_data.head()

[4]:
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0

```


[5]: # checking the shape of the dataset
car_data.shape

[5]: (8128, 13)

[6]: # basic info
```

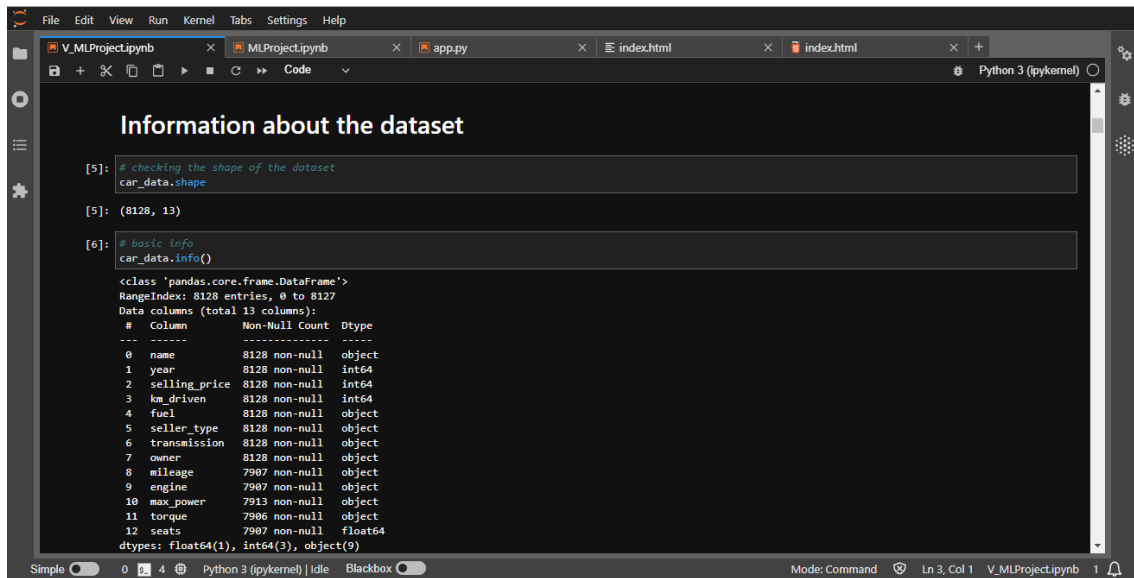
Figure 6: upload data

## 4 Data Preprocessing

Preprocessing the data is an essential step before training a machine learning model. Here are the steps that I have taken to preprocess the data for the used car price prediction dataset:

### 4.1 collect of information:

- 1- we have to check the shape of the dataset.
- 2- get some basic information like the type of each value.
- 3- then we have to show some details about the data.
- 4- we show the number of duplicated values.



```

[5]: # checking the shape of the dataset
car_data.shape

[5]: (8128, 13)

[6]: # basic info
car_data.info()

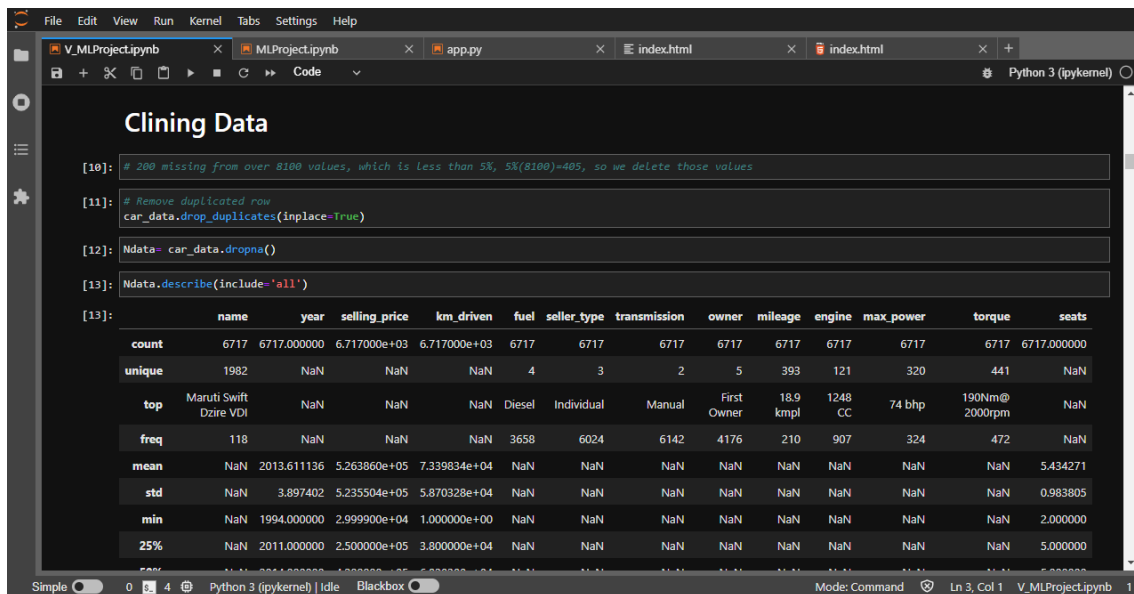
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
#   Column      Non-Null Count  Dtype
---  -
0    name        8128 non-null   object
1    year        8128 non-null   int64
2    selling_price 8128 non-null   int64
3    km_driven   8128 non-null   int64
4    fuel        8128 non-null   object
5    seller_type 8128 non-null   object
6    transmission 8128 non-null   object
7    owner       8128 non-null   object
8    mileage     7907 non-null   object
9    engine      7907 non-null   object
10   max_power   7913 non-null   object
11   torque      7906 non-null   float64
12   seats       7907 non-null   object
dtypes: float64(1), int64(3), object(9)

```

Figure 7: information about our data

## 4.2 Handling Missing Values:

first, we had checking the null values in the dataset. and we get only 200 missing from over 8100 values, which is less than 5%,  
 $(5\%(8100)=405)$ , so we delete those values.



```

[10]: # 200 missing from over 8100 values, which is less than 5%, 5%(8100)=405, so we delete those values

[11]: # Remove duplicated row
car_data.drop_duplicates(inplace=True)

[12]: Ndata= car_data.dropna()

[13]: Ndata.describe(include='all')

[13]:
   name      year  selling_price  km_driven  fuel  seller_type  transmission  owner  mileage  engine  max_power  torque  seats
count  6717  6717.000000  6.717000e+03  6.717000e+03  6717  6717  6717  6717  6717  6717  6717  6717  6717.000000
unique   1982         NaN         NaN         NaN         4         3         2         5      393      121      320      441         NaN
top  Maruti Swift  Dzire VDI         NaN         NaN         NaN  Diesel  Individual  Manual  First Owner  18.9 kmpl  1248 CC  74 bhp  190Nm@ 2000rpm  NaN
freq    118         NaN         NaN         NaN  3658  6024  6142  4176  210  907  324  472         NaN
mean  NaN  2013.611136  5.263860e+05  7.339834e+04  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  5.434271
std  NaN  3.897402  5.235504e+05  5.870328e+04  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  0.983805
min  NaN  1994.000000  2.999900e+04  1.000000e+00  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  2.000000
25%  NaN  2011.000000  2.500000e+05  3.800000e+04  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  5.000000

```

Figure 8: cleaning the data

## 4.3 Encoding Categorical Variables:

we Identify categorical variables in the dataset, such as car seller type, fuel type, or transmission type...  
 we choose Ordinal Encoding (Encode categories with an arbitrary numeric order if there is an inherent order.) as an appropriate

encoding method based on the number of unique categories and the nature of the variable.

```

[14]: Ndata['seller_type'] = Ndata['seller_type'].map({'Individual':0, 'Dealer':1, 'Trustmark Dealer':2})
[15]: Ndata['transmission'] = Ndata['transmission'].map({'Manual':0, 'Automatic':1})
[16]: Ndata['owner'] = Ndata['owner'].map({'First Owner':0, 'Second Owner':1, 'Third Owner':2, 'Fourth & Above Owner':3, 'Test Drive Car':4})
[17]: Ndata['fuel'] = Ndata['fuel'].map({'Diesel':0, 'Petrol':1, 'CNG':2, 'LPG':3})
[18]: Ndata
[18]:
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	0	0	0	0	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	0	0	0	1	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	1	0	0	2	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	0	0	0	0	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	1	0	0	0	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0

Figure 9: changing categorical features to numerical

#### 4.4 remove the suffix:

we remove the suffix of some features to have a float type of variables.

```

[19]: # Extract numeric part and convert 'mileage' column to float
Ndata['mileage'] = Ndata['mileage'].apply(lambda x: re.findall(r'\d+\.\d+', x)[0]).astype(float)
# Check the resulting data type
result_type = Ndata['mileage'].dtype
print(result_type)

float64

[20]: # Extract numeric part and convert 'engine' column to float
Ndata['engine'] = Ndata['engine'].apply(lambda x: re.findall(r'\d+\.\d+', x)[0]).astype(float)

[21]: # Extract numeric part and convert 'max_power' column to float
Ndata['max_power'] = Ndata['max_power'].apply(lambda x: re.findall(r'\d+\.\d+', x)[0]).astype(float)

[22]: Ndata.head()
[22]:
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	0	0	0	0	23.40	1248.0	74.00	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	0	0	0	1	21.14	1498.0	103.52	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	1	0	0	2	17.70	1497.0	78.00	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	0	0	0	0	23.00	1396.0	90.00	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	1	0	0	0	16.10	1298.0	88.20	11.5@ 4,500(kgm@ rpm)	5.0

Figure 10: remove the suffix

## 5 Model Training

### 5.1 Data Exploration:

before modeling we should do some visualization about the relation between the features, to decide which is the best choice for the

training model.

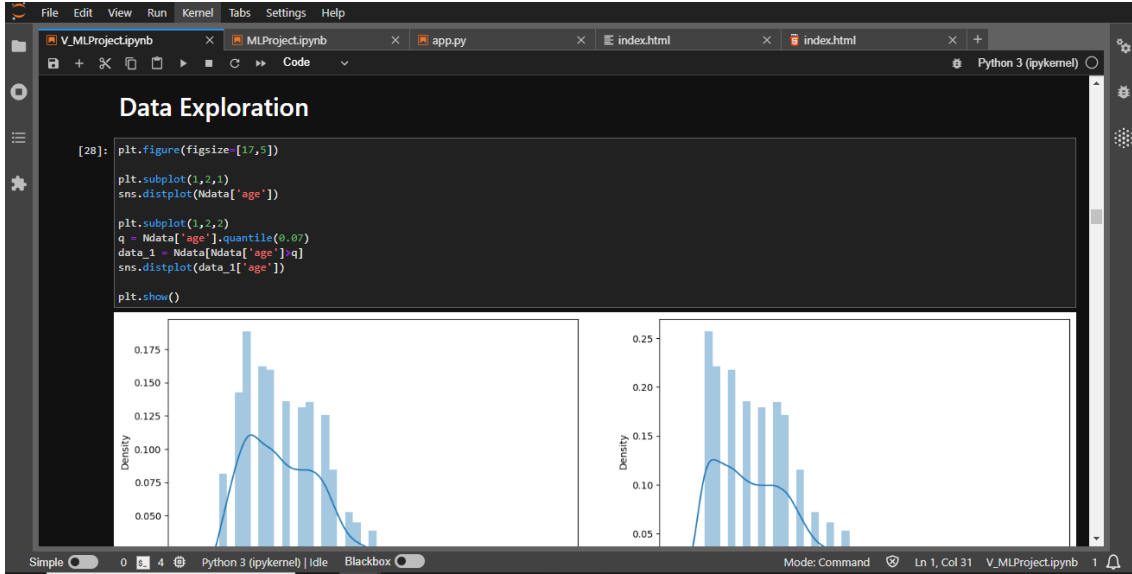


Figure 11: distribution of age

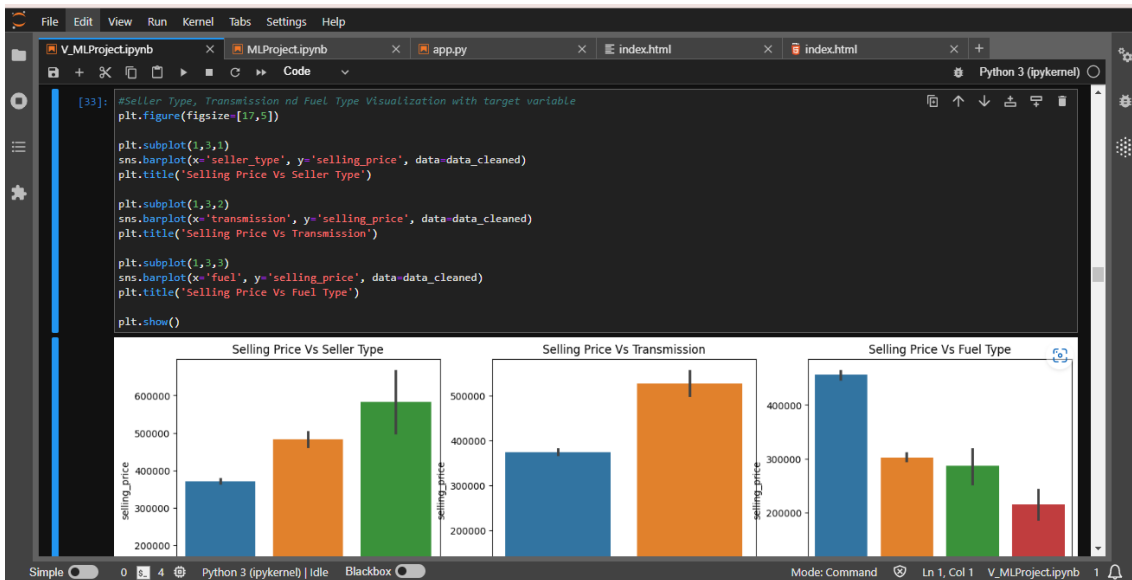


Figure 12: distribution of selling\_price vs other

## 5.2 Splitting the Data:

in this step, we do the split of data to train some models like Linear Regression, Polynomial Regression, Ridge Regression, Lasso Regression, and ElasticNetCV Regression.

```

[37]: # Splitting the data to Numerical Features and string Features
num_features = list(data_cleaned.select_dtypes('number'))
num_features.remove('selling_price')

str_features = list(data_cleaned.select_dtypes('string'))

print(f'The Numerical Features: {num_features}')
print(f'The String Features: {str_features}')

The Numerical Features: ['age', 'km_driven', 'fuel', 'seller_type', 'transmission', 'owner', 'mileage', 'engine', 'max_power', 'seats']
The String Features: ['CompanyName', 'torque']

[38]: # taking a copy from data
data_modeling = data_cleaned.copy()

# define dataset
X, y = data_modeling.drop("selling_price", axis=1), data_modeling["selling_price"]

# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

[39]: from sklearn.pipeline import Pipeline
from sklearn.pipeline import make_pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder

```

Figure 13: Split the data

## 6 Evaluation Metrics

Here we compare the result of the models to choose the better (who has the higher r2 score in the training a test modeling).

	Training Accuracy	Teasting Accuracy
<b>Linear Regression</b>	87.79	83.21
<b>Polynomial Regression</b>	94.11	-8288659.89
<b>Ridge Regression</b>	85.77	78.00
<b>Lasso Regression</b>	93.52	76.75
<b>ElasticNetCV Regression</b>	12.45	13.13

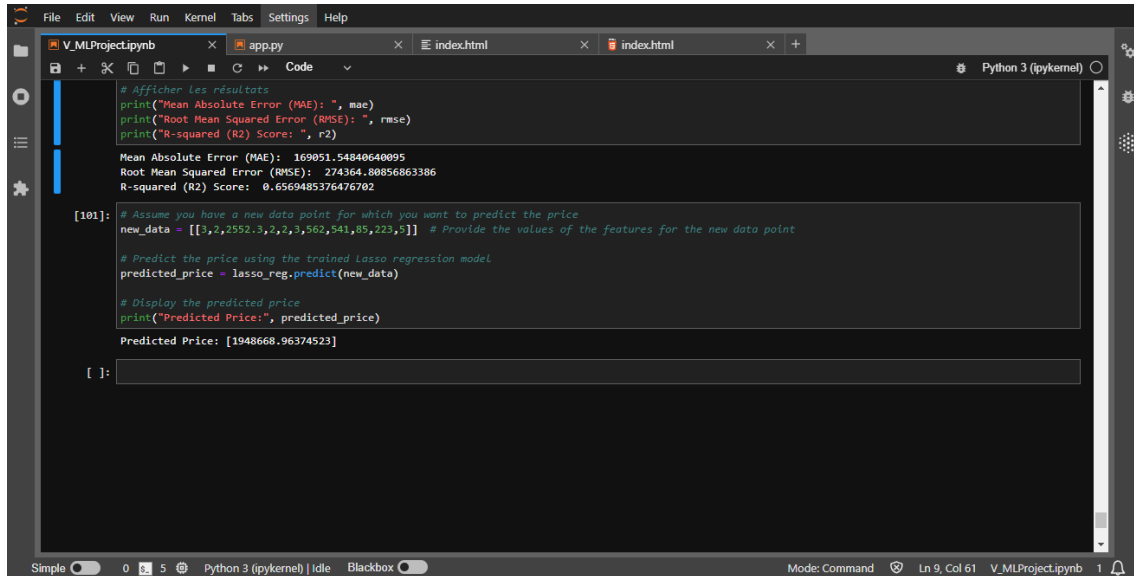
Figure 14: Models summarization

that's why we choose the model Lasso Regression which has 93.52 in Training Accuracy and 76.75 in Testing Accuracy.

## 7 Results and Discussion

### 7.1 the result of prediction

we get as a result the price prediction of cars like you see in this photo



```
File Edit View Run Kernel Tabs Settings Help
V_MLProjectipynb x app.py x index.html x index.html Python 3 (pykernel)

# Afficher les résultats
print("Mean Absolute Error (MAE): ", mae)
print("Root Mean Squared Error (RMSE): ", rmse)
print("R-squared (R2) Score: ", r2)

Mean Absolute Error (MAE): 169051.54840640095
Root Mean Squared Error (RMSE): 274364.80856863386
R-squared (R2) Score: 0.6569485376476702

[101]: # Assume you have a new data point for which you want to predict the price
new_data = [[3,2,2552.3,2,2,3,562,541,85,223,5]] # Provide the values of the features for the new data point

# Predict the price using the trained Lasso regression model
predicted_price = lasso_reg.predict(new_data)

# Display the predicted price
print("Predicted Price:", predicted_price)

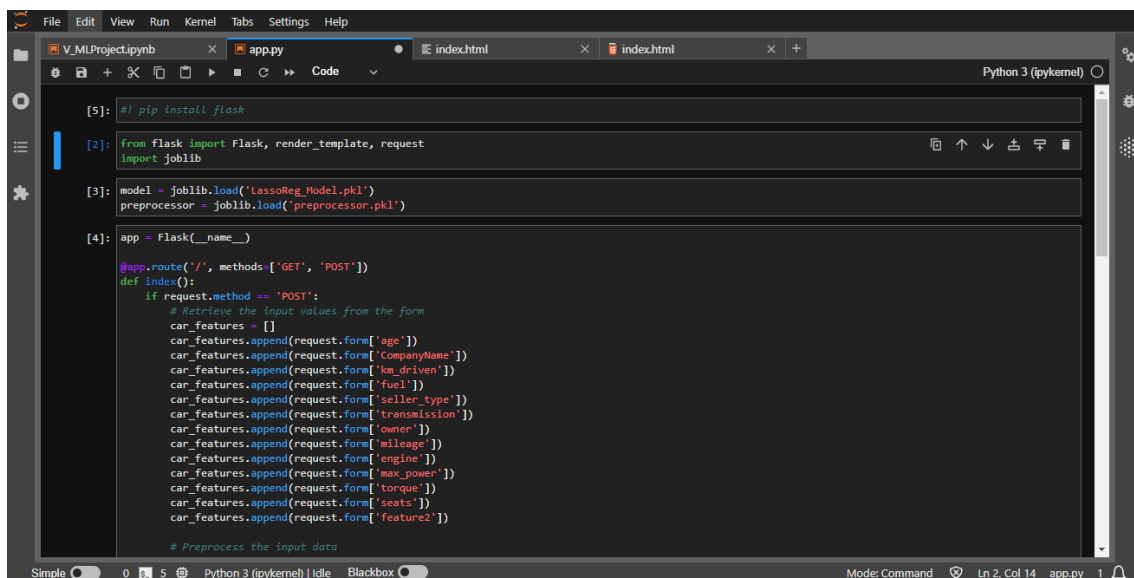
Predicted Price: [1948668.96374523]

[ ]:
```

Figure 15: the price prediction

### 7.2 connect the code of Jupyter with Flask file:

we used Flask to connect the code of the Model to a Web page



```
File Edit View Run Kernel Tabs Settings Help
V_MLProjectipynb x app.py x index.html x index.html Python 3 (pykernel)

[5]: #! pip install flask

[2]: from flask import Flask, render_template, request
import joblib

[3]: model = joblib.load('LassoReg_Model.pkl')
preprocessor = joblib.load('preprocessor.pkl')

[4]: app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        # Retrieve the input values from the form
        car_features = []
        car_features.append(request.form['age'])
        car_features.append(request.form['Company Name'])
        car_features.append(request.form['km driven'])
        car_features.append(request.form['fuel'])
        car_features.append(request.form['seller type'])
        car_features.append(request.form['transmission'])
        car_features.append(request.form['owner'])
        car_features.append(request.form['mileage'])
        car_features.append(request.form['engine'])
        car_features.append(request.form['max power'])
        car_features.append(request.form['torque'])
        car_features.append(request.form['seats'])
        car_features.append(request.form['feature2'])

        # Preprocess the input data
```

Figure 16: the flask code

```

car_features.append(request.form['feature2'])

# Preprocess the input data
transformed_features = preprocessor.transform([car_features])

# Make predictions
predicted_price = model.predict(transformed_features)[0]

# Render the HTML page with the predicted price
return render_template('index.html', predicted_price=predicted_price)

# Render the initial HTML page with the form
return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)

* Serving Flask app '__main__'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
C:\Users\megdiche leila\AppData\Local\Programs\Python\Python39\lib\site-packages\IPython\core\interactiveshell.py:3468: UserWarning: To exit: use 'exit',
'quit', or Ctrl-D.
warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)

```

Figure 17: the link to the navigator

### 7.3 the web page:

this is our view of the application's web

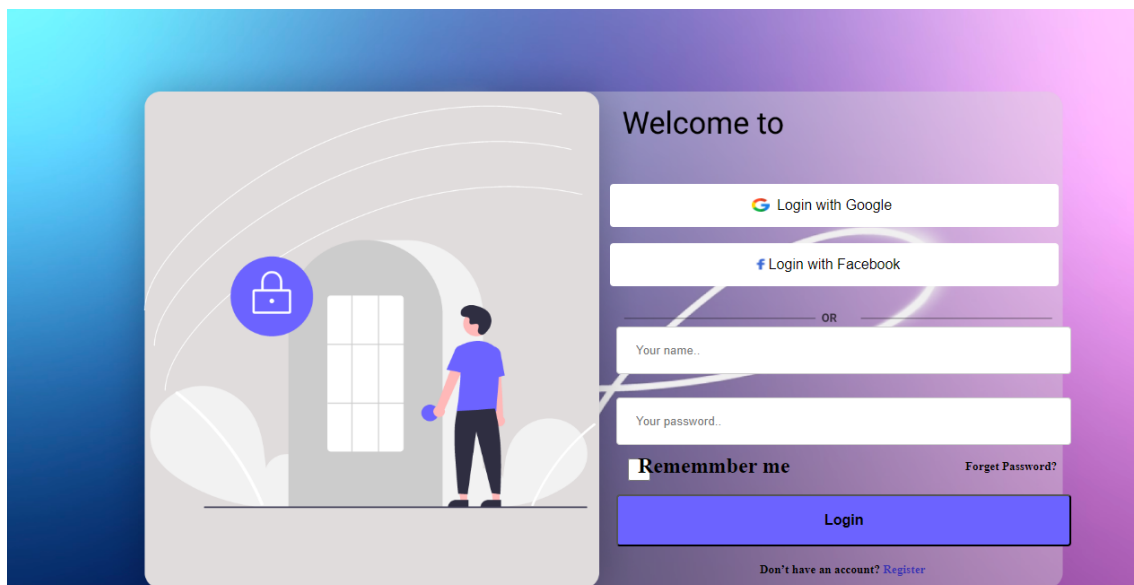


Figure 18: the first interface of app

## 8 Conclusion

The used car price prediction application yielded several key findings and insights:

**Model Performance:** The chosen Lasso regression model demonstrated good performance in predicting car prices. It effectively captured the relationships between the independent variables (features) and the target variable (car price). The model's

accuracy and precision were evaluated using appropriate metrics such as mean squared error or R-squared.

**Feature Importance:** The analysis revealed that certain features had a significant impact on car prices. These features could include factors like car age, mileage, engine size, and brand. Understanding the importance of these features can help car buyers and sellers make informed decisions and negotiate prices effectively.

**Strengths of the Model:** The Lasso regression model offers several strengths. It performs feature selection by shrinking less important features' coefficients to zero, allowing for a more interpretable and simplified model. It can handle a large number of features and automatically performs regularization to prevent overfitting.

**Limitations and Areas for Improvement:** Despite its strengths, the Lasso regression model and the used car price prediction application have some limitations. The model assumes a linear relationship between the features and the target variable, which may not always hold true. Additionally, the model's performance may be influenced by outliers, missing data, or multicollinearity among features. Addressing these issues and exploring other regression algorithms or advanced techniques could enhance the model's accuracy and robustness.

**Benefits in the Automotive Industry:** Accurately predicting car prices can benefit both car buyers and sellers. Buyers can use the model to estimate the fair market value of a used car and negotiate better deals. Sellers can set competitive prices based on market trends and demand, leading to faster sales. The model's relevance extends to various stakeholders in the automotive industry, including dealerships, insurers, and financial institutions, aiding in decision-making processes related to pricing, valuation, and risk assessment.

In conclusion, the used car price prediction application based on the Lasso regression model provides valuable insights and benefits for car buyers, sellers, and other industry players. While the model exhibits strengths, it is important to address its limitations and explore further improvements to enhance its performance and applicability in real-world scenarios

## 9 Upgrades to our project:

we look to upgrade our project to have a website and a mobile application that makes all the buyers and sellers in a direct



relationship and have an idea about all the cars in the market with a simple tap. we use Azure to have an idea about our application.

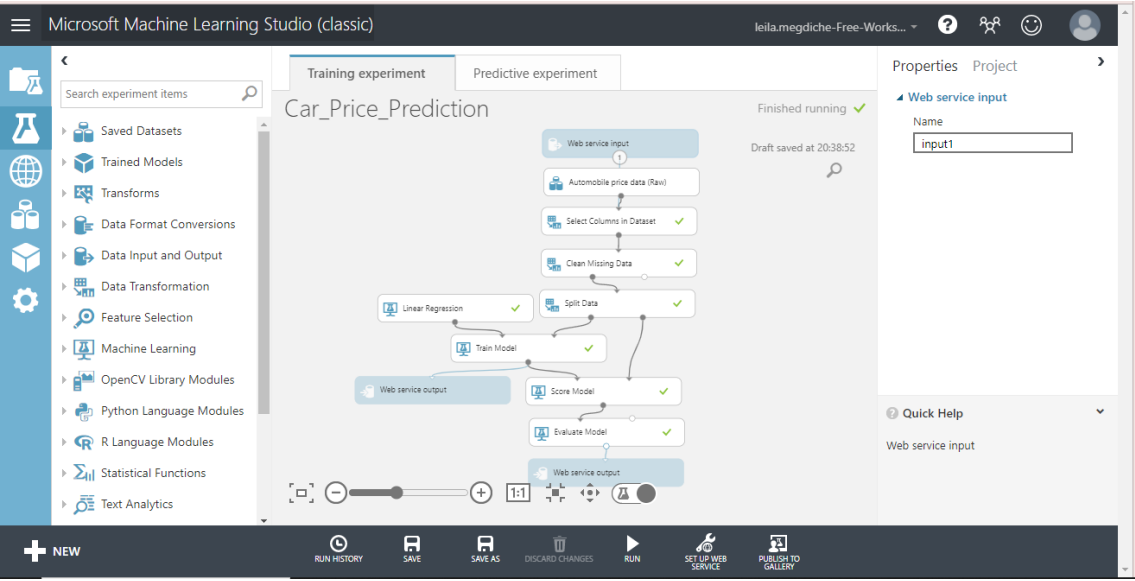


Figure 19: the model on azure

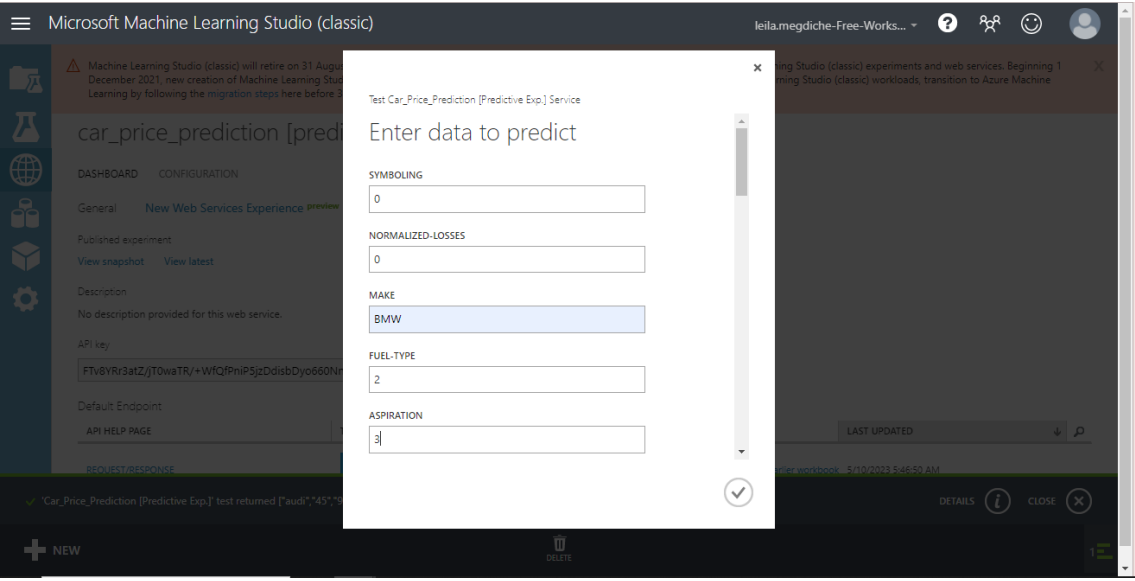


Figure 20: the formula of application on azure

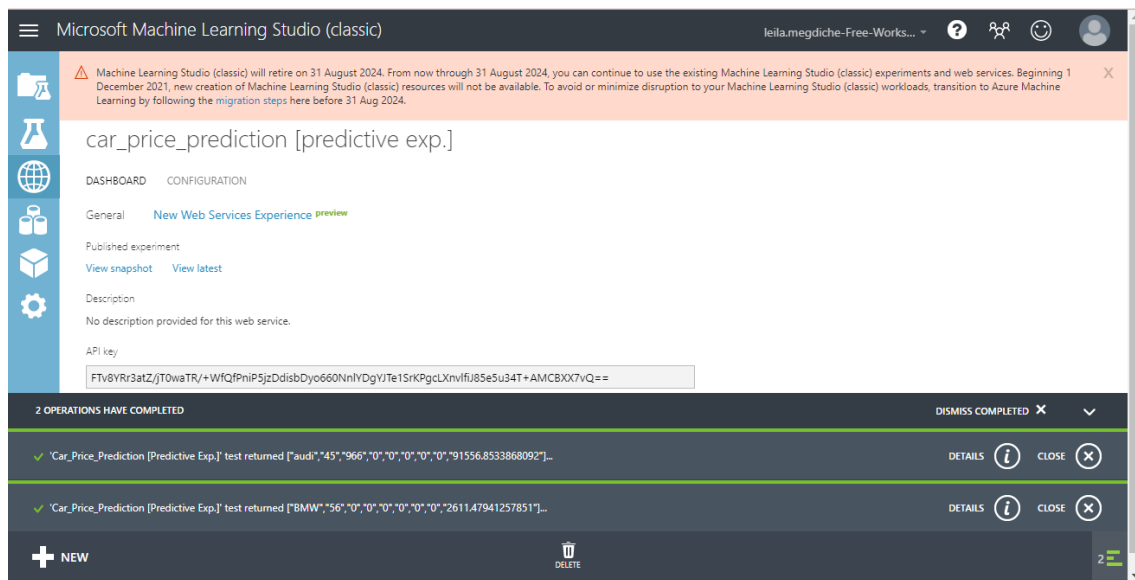


Figure 21: the price prediction result

## 10 References

- ✿ we get the dataset from KAGGLE and this is the link  
<https://www.kaggle.com/datasets/ishaanthareja007/car-details>.
- ✿ you can find all the codes in this link:  
[https://github.com/leila-megdiche/Used\\_cars\\_Price\\_prediction\\_Application](https://github.com/leila-megdiche/Used_cars_Price_prediction_Application)