2021-2022 SPRING SEMESTER

CSE443 MACHINE LEARNING LECTURE PROJECT-1

In this project you need to apply all seen topics in our Lecture. You need to submit all related files with your professor by sending a mail to o.sahingoz@iku.edu.tr **(sharing a cloud folder)**

Each dataset can only be used for only one user. (First submitted students is accepted, the other(s) should change the dataset)

Upper Limit of the Project is greater than 100. Writing a paper is optional. If you write and submit the paper, you will get +50 Points (up to).

a) Find a dataset from Internet with at least 10.000 data in it.

- Kaggle **https://www.kaggle.com/datasets**
- UCI Machine Learning Repository **https://archive.ics.uci.edu/ml/datasets.php**
- **https://www.v7labs.com/blog/best-free-datasets-for-machine-learning**
- https://imerit.net/blog/the-60-best-free-datasets-for-machine-learning-all-pbm/
- Google Dataset Search:
- CMU Libraries: Discover high-quality datasets thanks to the collection of Huajin Wang, at CMU

b) Show related information about the dataset. (How many records does it have? What are the features? Types of the features?.... etc.)

(15 Points)

- Dataset should contain at least 15 features in it.

DATASET NAME : Car Features and MSRP

DATASET WEB LINK : https://www.kaggle.com/datasets/CooperUnion/cardataset

DATASET INFO

How many records does it contains : 11914

How many features does it have : 16

How many different classes exist in the dataset? : 3

What are number of examples for each classes? : 8 object, 5 int, 2 float

How many NULL values exist? (depending on the features distinctly) :

```python
# checking null values
df.isnull().sum()
```

```
Make                  0
Model                 0
Year                  0
Engine Fuel Type      3
Engine HP            69
Engine Cylinders     30
Transmission Type     0
Driven_Wheels         0
Number of Doors       6
Market Category    3092
Vehicle Size          0
Vehicle Style         0
highway MPG           0
city mpg              0
Popularity            0
MSRP                  0
dtype: int64
```

Which features are not numeric?: “Make”, “Model”, “Engine Fuel Type”, “Transmission Type”, “Driven_Wheels”, “Market Category”, “Vehicle Size”, “Vehicle Style”

c) Use **Label Encoding** for at least one of the features (Explain your reason “why do make this operation?”) (10 Points)

Machines understand numbers, not text. We need to convert each text category to numbers in order for the machine to process them using mathematical equations. For this reason we can use label encoding or one hot encoding.

Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

We apply Label Encoding when:

1. The categorical feature is ordinal (like Jr. kg, Sr. kg, Primary school, high school)

2. The number of categories is quite large as one-hot encoding can lead to high memory consumption

I'm going to use Label encoding for 'Make', 'Model' 'Engine Fuel Type', 'Transmission Type', 'Driven_Wheels', and 'Vehicle Style' columns because there are many unique values and one-hot encoding can lead to high memory consumption.

```python
# Label encoding
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
df['Make']= label_encoder.fit_transform(df['Make'])
print(df['Make'].unique())
df['Model']= label_encoder.fit_transform(df['Model'])
print(df['Model'].unique())
df['Engine Fuel Type']= label_encoder.fit_transform(df['Engine Fuel Type'])
print(df['Engine Fuel Type'].unique())
df['Transmission Type']= label_encoder.fit_transform(df['Transmission Type'])
print(df['Transmission Type'].unique())
df['Driven_Wheels']= label_encoder.fit_transform(df['Driven_Wheels'])
print(df['Driven_Wheels'].unique())
df['Vehicle Style']= label_encoder.fit_transform(df['Vehicle Style'])
print(df['Vehicle Style'].unique())
```

Python

```
Output exceeds the size limit. Open the full output data in a text editor
[ 4  3 12 31 10 33 47 29 32 13  1 45 30 28 36 37 39 15 19 35 18 34 43 14
  8 21  5  9 11 22 25 42 46 41  7  0 38 27 24  2 23 26 20 40 16 17 44  6]
[ 1  0  2  3  4  6  7  8 10  9  5 13 12 15 16 14 17 18
  19 20 21 22 23 11 25 24 26 27 28 29 30 31 34 33 36 39
```

d) Use **One Hot encoding** for at least one of the features (Explain your reason “why do make this operation?”) (10 Points)

We apply One-Hot Encoding when:

1. The categorical feature is not ordinal (like the countries above)

2. The number of categorical features is less so one-hot encoding can be effectively applied

```python
# One-hot encoding
cat_features = ["Vehicle Size"]
df = pd.get_dummies(df, columns = cat_features)
df.info()
```

Python

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10582 entries, 0 to 11913
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Make                  10582 non-null  int32
 1   Model                 10582 non-null  int32
 2   Year                  10582 non-null  int64
 3   Engine Fuel Type      10582 non-null  int32
 4   Engine HP             10582 non-null  float64
 5   Engine Cylinders      10582 non-null  float64
 6   Transmission Type     10582 non-null  int32
 7   Driven_Wheels         10582 non-null  int32
 8   Number of Doors       10576 non-null  float64
 9   Vehicle Style         10582 non-null  int32
 10  highway MPG           10582 non-null  int64
 11  city mpg              10582 non-null  int64
 12  Popularity            10582 non-null  int64
 13  MSRP                  10582 non-null  int64
 14  Vehicle Size_Compact  10582 non-null  uint8
 15  Vehicle Size_Large    10582 non-null  uint8
 16  Vehicle Size_Midsize  10582 non-null  uint8
dtypes: float64(3), int32(6), int64(5), uint8(3)
memory usage: 1023.1 KB
```

e) Analyze the Missing Values

a. **Delete some columns** (Explain your reason "why do make this operation?")

(10 Points)

To be able to generate the best prediction results, some modifications on the dataset are required. Deleting some columns is one of the very common operations. We delete columns when they are not really correlated to the feature we want to predict or when the column has too many NaN values.

The handling of missing data is very important during the preprocessing of the dataset as many machine learning algorithms do not support missing values.

In this case, we observe, Market Category has the highest missing values followed by Engine HP and Enginer Cylinders.
We see it is really not that useful for our model, especially when our data already consists of additional details like 'Model', 'Vechicle Style', 'Vechicle Size or Make'. Anyone can which type of car it is from these features. We will drop the entire column 'Market Category'.

```python
df = df.drop('Market Category', axis = 1)
# dropping 'market category' column as MSRP is independent of it and hence not useful in predicting price of car.
```

Python

b. **Delete some rows** (Explain your reason "why do make this operation?") (10 Points)

Deleting rows is also a very common operation. The rows which are having one or more columns values as null can be dropped. I also removed duplicates.

```python
df.drop(df[df['MSRP'] == 0].index,inplace=True)
# Dropping rows which have zero as a value for MSRP column as it is our dependent/target variable.
```

Python

```python
df.duplicated().sum()
#checking for any duplicates in the data
```

Python

```
715
```

```python
df.drop_duplicates(keep=False,inplace=True)
#removing the duplicates in the data
```

Python

c. **Impute** some missing data (Explain your reason "why do make this operation?") (10 Points)

To deal with the NaN values we can erase the rows that have NaN values. But this is not a good choice because in such a way we lose the information, especially when we work with small datasets. So we impute NaN values with specific methods or values.

```python
df['Engine HP'] = df['Engine HP'].fillna(0)
df['Engine Cylinders'] = df['Engine Cylinders'].fillna(0)
# Replaced them with zero because electric vehicles don't have engine.
df['Number of Doors'] = df['Number of Doors'].fillna(df['Number of Doors'].mode()[0])
df['Engine Fuel Type'] = df['Engine Fuel Type'].fillna('regular unleaded')
```

✓ 0.3s Python

```python
df.isnull().sum()
# No more missing values.
```

✓ 0.4s Python

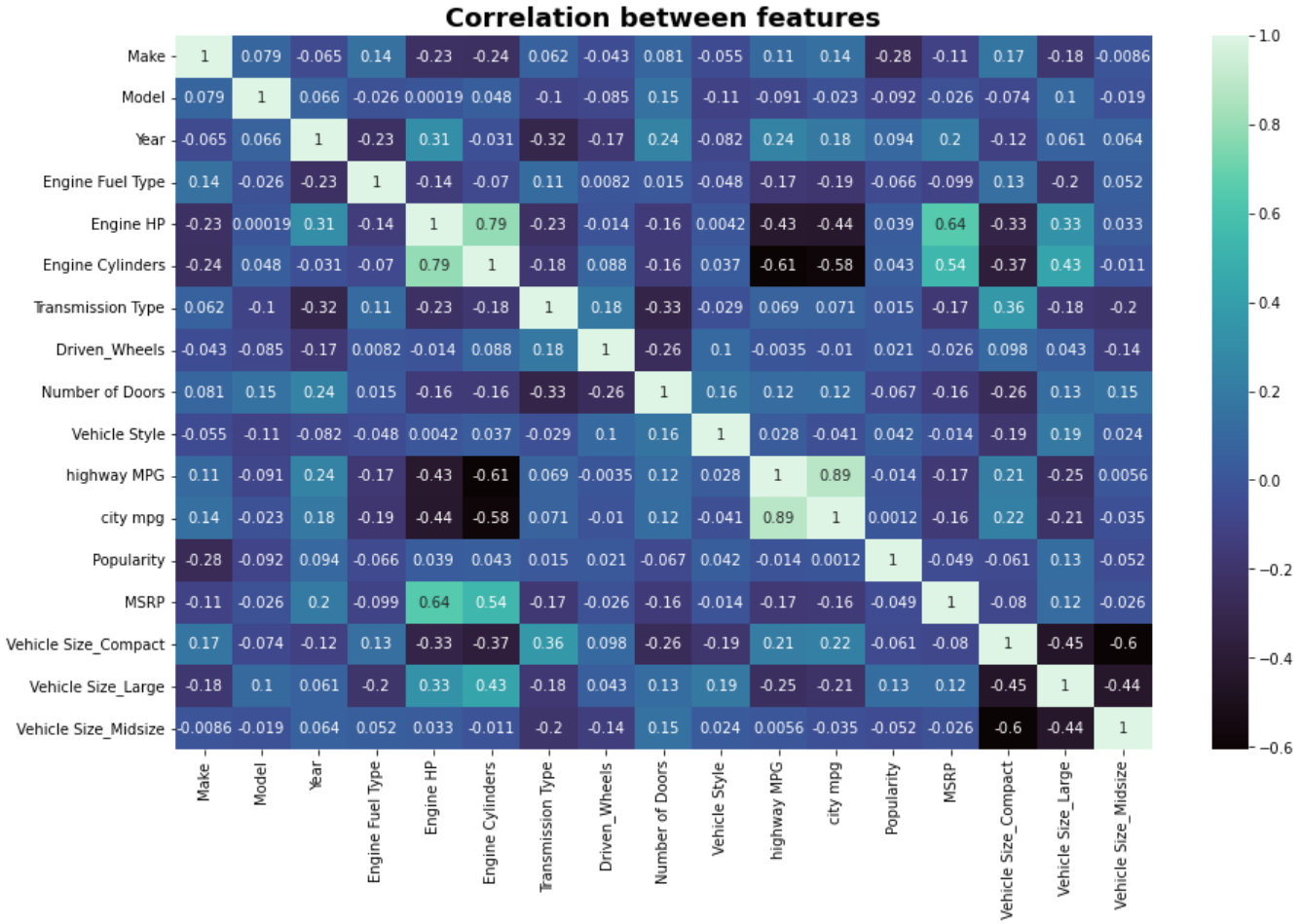
```
Make                0
Model               0
Year                0
Engine Fuel Type    0
Engine HP           0
Engine Cylinders    0
Transmission Type   0
Driven_Wheels       0
Number of Doors     0
Vehicle Size        0
Vehicle Style       0
highway MPG         0
city mpg            0
Popularity          0
MSRP                0
dtype: int64
```

f) Find the best correlated Features in the Dataset (10 Points)

```
# df.corr()
df.corrwith(df['MSRP'])
```

✓ 0.6s Python

```
Make                      -0.107882
Model                     -0.026424
Year                       0.195554
Engine Fuel Type          -0.099380
Engine HP                  0.644059
Engine Cylinders           0.537270
Transmission Type         -0.173479
Driven_Wheels             -0.026134
Number of Doors           -0.158659
Vehicle Style             -0.014202
highway MPG               -0.169020
city mpg                  -0.164099
Popularity                -0.049125
MSRP                       1.000000
Vehicle Size_Compact      -0.080227
Vehicle Size_Large         0.119835
Vehicle Size_Midsize      -0.026367
dtype: float64
```



The best correlated features with the target feature are: “Engine Cylinders” and “Engine HP”.

g) Execute a Normalization/Scaling in the Dataset (10 Points)

```
X
```

✓ 0.8s Python

```
array([[4.000e+00, 1.000e+00, 2.011e+03, ..., 1.000e+00, 0.000e+00,
        0.000e+00],
       [4.000e+00, 0.000e+00, 2.011e+03, ..., 1.000e+00, 0.000e+00,
        0.000e+00],
       [4.000e+00, 0.000e+00, 2.011e+03, ..., 1.000e+00, 0.000e+00,
        0.000e+00],
       ...,
       [0.000e+00, 8.830e+02, 2.012e+03, ..., 0.000e+00, 0.000e+00,
        1.000e+00],
       [0.000e+00, 8.830e+02, 2.013e+03, ..., 0.000e+00, 0.000e+00,
        1.000e+00],
       [2.500e+01, 8.840e+02, 2.006e+03, ..., 0.000e+00, 0.000e+00,
        1.000e+00]])
```

```
# normalizing the data
from sklearn.preprocessing import StandardScaler
sc_x=StandardScaler()
sc_y=StandardScaler()

X=sc_x.fit_transform(X)
```

```
X
```

✓ 0.2s Python

```
array([[-1.29974219, -1.69922939,  0.00824523, ...,  1.26668048,
        -0.57393117, -0.76350959],
       [-1.29974219, -1.70300752,  0.00824523, ...,  1.26668048,
        -0.57393117, -0.76350959],
       [-1.29974219, -1.70300752,  0.00824523, ...,  1.26668048,
        -0.57393117, -0.76350959],
       ...,
       [-1.58246154,  1.63308386,  0.1517502 , ..., -0.78946507,
        -0.57393117,  1.30974124],
       [-1.58246154,  1.63308386,  0.29525517, ..., -0.78946507,
        -0.57393117,  1.30974124],
       [ 0.18453435,  1.63686199, -0.70927961, ..., -0.78946507,
        -0.57393117,  1.30974124]])
```

h) Train your new dataset **at least 5 different Machine Learning algorithms** (15 Points)

THE PREFERRED ALGORITHMS ARE

- LGBM Regressor
- Random Forest Regressor
- XGB Regressor
- SVM
- Linear Regression

i) Use **5-fold** approach to measure the performance of the system

(10 Points)

In k-fold cross-validation, the entire dataset is divided into k subsamples to repeat the model training and validation process k times. Each subsample is a validation set in one round and a part of the training set in the rest of the rounds.

WITH A RANDOM SELECTION WE REACHED THE FOLLOWING RESULTS

SVM:

Training Accuracy : -0.03343647679638839

Testing Accuracy : -0.04914574580006548

Linear Regression:

Training Accuracy : 0.4816464010542366

Testing Accuracy : 0.5425761928514978

LGBM Regressor:

Training Accuracy : 0.9417074955054746

Testing Accuracy : 0.9453866858564184

XGB Regressor:

Training Accuracy : 0.9937978820180313

Testing Accuracy : 0.9721870241027762

Random Forest:

Training Accuracy : 0.9860581511326963

Testing Accuracy : 0.9751756180436367

WITH 5-FOLD APPROACH I REACHED THE FOLLOWING RESULTS

SVM:

Mean score (std): -0.035 (0.006)

Linear Regression:

Mean score (std): 0.512 (0.080)

LGBM Regressor:

Mean score (std): 0.905 (0.084)

XGB Regressor:

Mean score (std): 0.933 (0.079)

Random Forest:

Mean score (std): 0.935 (0.083)

So, the best suitable algorithm for the given dataset is "Random Forest" with a score of "93.5%".

j) Put their results to **a table** to make a comparison (5 Points)

	Model	Training Accuracy %	Testing Accuracy %	MAPE	r2 score	MSE	RSME	Time
0	Support Vector Machine	-0.033436	-0.049146	1.635443	-0.049146	2.608926e+09	51077.646645	22.863063
1	Linear Regression	0.481646	0.542576	1.298923	0.542576	1.137482e+09	33726.583504	1.581266
2	LGBM Regressor	0.941707	0.945387	0.133840	0.945387	1.358077e+08	11653.657052	3.168372
3	XGB Regressor	0.993798	0.972187	0.098800	0.972187	6.916293e+07	8316.425418	12.218570
4	Random Forest Regressor	0.986058	0.975176	0.096304	0.975176	6.173115e+07	7856.917443	6.179033

- Evaluating the model accuracy is an essential part of the process in creating machine learning models to describe how well the model is performing in its predictions. The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics. It usually expresses the accuracy as a ratio.
- MSE (Mean Squared Error) represents the difference between the original and predicted values extracted by squared the average difference over the data set.
- RMSE (Root Mean Squared Error) is the error rate by the square root of MSE.
- R-squared (Coefficient of determination) represents the coefficient of how well the values fit compared to the original values. The value from 0 to 1 interpreted as percentages. The higher the value is, the better the model is.

k) Calculate **the training time** for all of them

(10 Points)

SVM: 22.86
Linear Regression: 1.58
LGBM Regressor: 3.17
XGB Regressor: 12.22
Random Forest: 6.18

l) Select the best 10 features from the database

(10 Points)

Feature selection is the process of reducing the number of input variables when developing a predictive model.
It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.
Statistical-based feature selection methods involve evaluating the relationship between each input variable and the target variable using statistics and selecting those input variables that have the strongest relationship with the target variable.

SHOW THE LIST OF THE FEATURES

Feature	**Correlation with MSRP**
Engine HP	0.644059
Engine Cylinders	0.537270
Year	0.195554
Transmission Type	-0.173479
highway MPG	-0.169020
city mpg	-0.164099
Number of Doors	-0.158659
Vehicle Size_Large	0.119835
Make	-0.107882
Engine Fuel Type	-0.099380

m) Write **a Conference paper** to Show all your reached results. **(OPTIONAL)** (50 Points)