Due: Final Exam.

END3971 Artificial Intelligence & Expert Systems

Assignment #4

Member 1 Number / Name SurName

Member 2 Number / Name SurName

Member 3 Number / Name SurName

Member 4 Number / Name SurName

1 Linear Regression Model with Regularization

Regularization is a process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting (Wikipedia). One way to regularize is to add a constraint to the loss function:

Regularized Loss = Loss Function + Constraint

There are multiple different forms of constraints that we could use to regularize. The three most popular ones are Ridge Regression, Lasso, and Elastic Net.

Ridge Regression

Ridge regression is also called **L2 regularization**. It adds a constraint that is a linear function of the squared coefficients.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2$$

Lasso

Lasso is also known as **L1 regularization**. It penalizes the model by the absolute weight coefficients.

$$\hat{\beta}^{\text{lasso}}=\arg\min_{\beta}\sum_{n=1}^{N}\frac{1}{2}(y_n-\beta x_n)^2+\lambda\sum_{i=1}^{p}|\beta_i|$$

Elastic Net

Elastic Net is the combination of the L1 regularization and L2 regularization. It can both shrink the coefficients as well as eliminate some of the insignificant coefficients.

$$\beta = \operatorname{argmin} \frac{1}{2}\left[\sum_{i=1}^{n}\left|\upsilon_r(y_i) - \sum_{j=1}^{k}\upsilon_j(x_{ij})\beta_j\right| + n\lambda_1\sum_{j=1}^{k} w_{s_j}|\beta_j| + \frac{n}{2}\lambda_2\sum_{j=1}^{k} w_{s_j}{\beta_j}^2\right]$$

For regularized regression, we choosed Elastic Net which combinde L1(Lasso) and L2(Ridge) regularization for regression with specs **alpha = 6, l1_ratio = 1, max_iter = 1000000**.

For metrics of this regularized regression Training Score, Test Score and RMSE(Root Meam Square Error) are considered.

Here is our results…

➔ Training score: 0.8734856272678422

➔Test score: 0.9001003227840508

➔RMSE: 3808.6031364157006

Here is part of code for elastic net regularized regression. When yo urun this code from your python compiler, you will get these results…

#imports libraries

```
# ======================================================================
import pandas as pd
import math
from sklearn.model_selection import train_test_split
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.cluster import KMeans
# ======================================================================
```

```
# import csv data from Data_3000.csv file
cars = pd.read_csv(r'.\Data_3000.csv', engine = 'python', sep = ';')

#Splitting motor(1.3,1.4,1.5...) from model column
# ================================================================
Motor = cars['model'].apply(lambda x : x.split(' ')[0])
cars.insert(3,"motor",Motor)
cars.drop(['model'], axis = 1, inplace = True)
cars["motor"] = cars["motor"].astype(float)
# ================================================================

# Drop ser and marka columns from dataset, because all same type
# ================================================================
cars = cars.drop('seri', axis = 1)
cars = cars.drop('marka', axis = 1)
# ================================================================

# dummification for categorical data (yakit, vites, cekis, renk, garanti, kimden)
cars_dum = pd.get_dummies(cars, prefix_sep='_', drop_first=True)

# split output data as y variable and drop it from dataset
y = cars_dum[["fiyat"]].to_numpy()
cars_dum = cars_dum.drop(columns = ["fiyat"])

# split input data as x varible and scale it with X
# use scaled data X for regularized regression
# use unscaled data x for k-mean
# ================================================================
x = cars_dum.values
columns = cars_dum.columns
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
```

```
cars_dum = pd.DataFrame(x_scaled)

cars_dum.columns = columns


X = cars_dum.to_numpy()

# ================================================================
```

 # Split train and test data for regularized regression

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=42)



print('\n')

print(">>>>>>>>>>> REGULARIZED REGRESSION START <<<<<<<<<<")

"""
```

ELASTIC NET REGULARIZED REGRESSION MODEL

For regularized regression, we choosed Elastic Net which combinde L1(Lasso) and L2(Ridge) regularization for regression.

```
"""

# ================================================================

elasticNet = ElasticNet(alpha = 6, l1_ratio = 1, max_iter = 1000000)

elasticNet.fit(X_train, y_train)

print('Training score: {}'.format(elasticNet.score(X_train, y_train)))

print('Test score: {}'.format(elasticNet.score(X_test, y_test)))

y_pred = elasticNet.predict(X_test)

mse = mean_squared_error(y_test, y_pred)

rmse = math.sqrt(mse)

print('RMSE: {}'.format(rmse))

# ================================================================

print(">>>>>>>>>>> REGULARIZED REGRESSION FINISHED <<<<<<<<<<")

print('\n')
```

Due: Final Exam.

2 Clustering Using k-means Algorithm

You are supposed to prepare your report in a way you devised. Please clear these sentences before you start writing.

K-means clustering is one of the simplest unsupervised machine learning algorithms. Here, we'll explore what it can do and work through a simple implementation in Python.

Some facts about k-means clustering:

1. K-means converges in a finite number of iterations. Since the algorithm iterates a function whose domain is a finite set, the iteration must eventually converge.
2. Compared to other clustering methods, the k-means clustering technique is fast and efficient in terms of its computational cost.
3. It's difficult to predict the optimal number of clusters or the value of k. To find the number of clusters, we need to run the k-means clustering algorithm for a range of k values and compare the results.

For clustering, we applied K-Mean clustering which is an unsupervised learning method.

For clsutering it is important to select number of cluster.

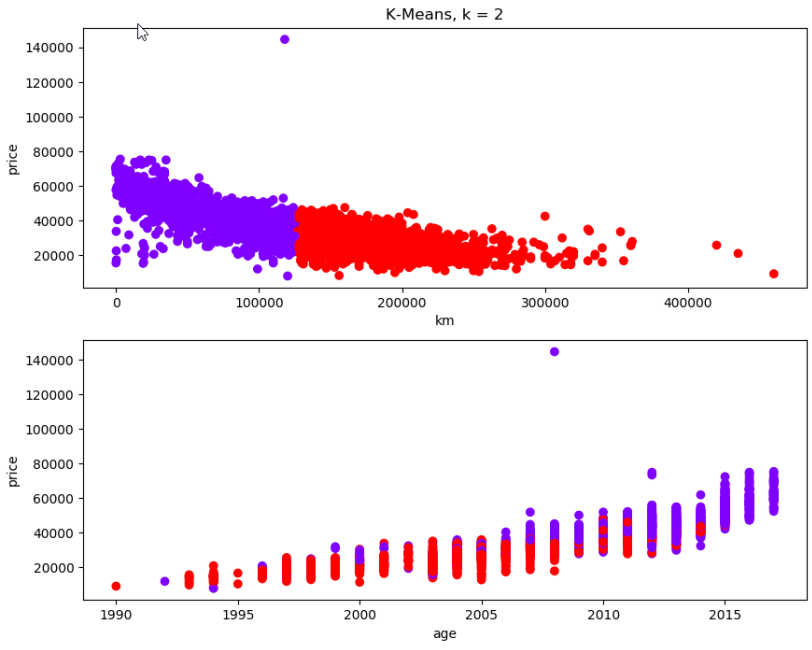
To select optimal number of cluster, we used Elbow Curve and decide optimal cluster number for this model. Unsclaled x data used for k-mean input.

Because this is an unsupervised method y data is not used.

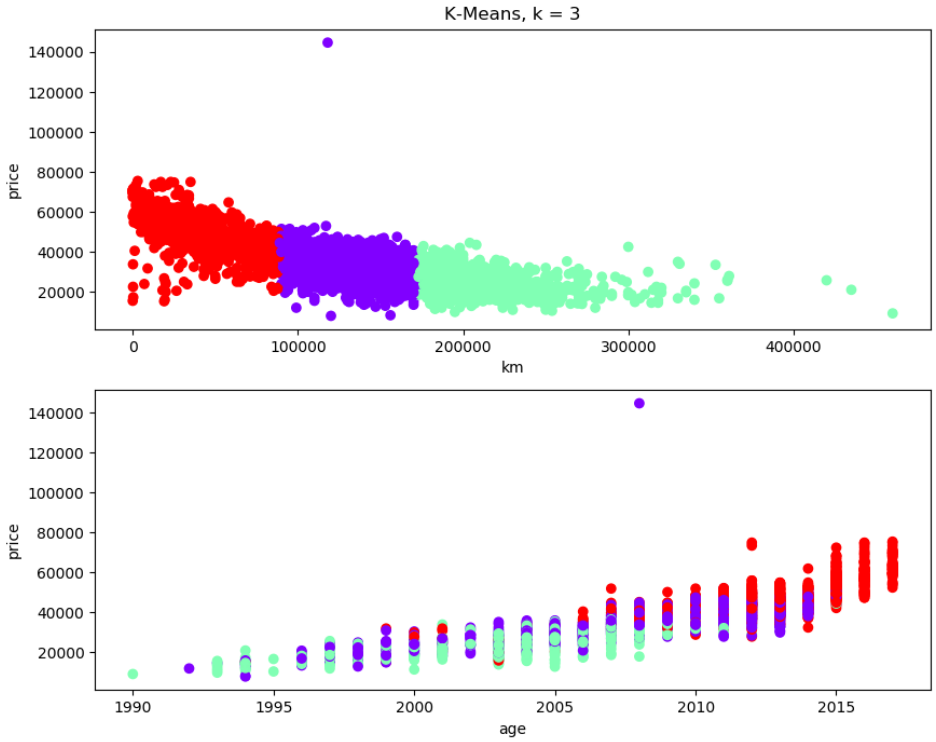
Here is our piece of code for K-Mean Clustering method with k = 2,3,4,5,6,7 and Price/Age, Price/km, Elbow Curve Plots…

Just run the code from your python compiler and you will see these plots.

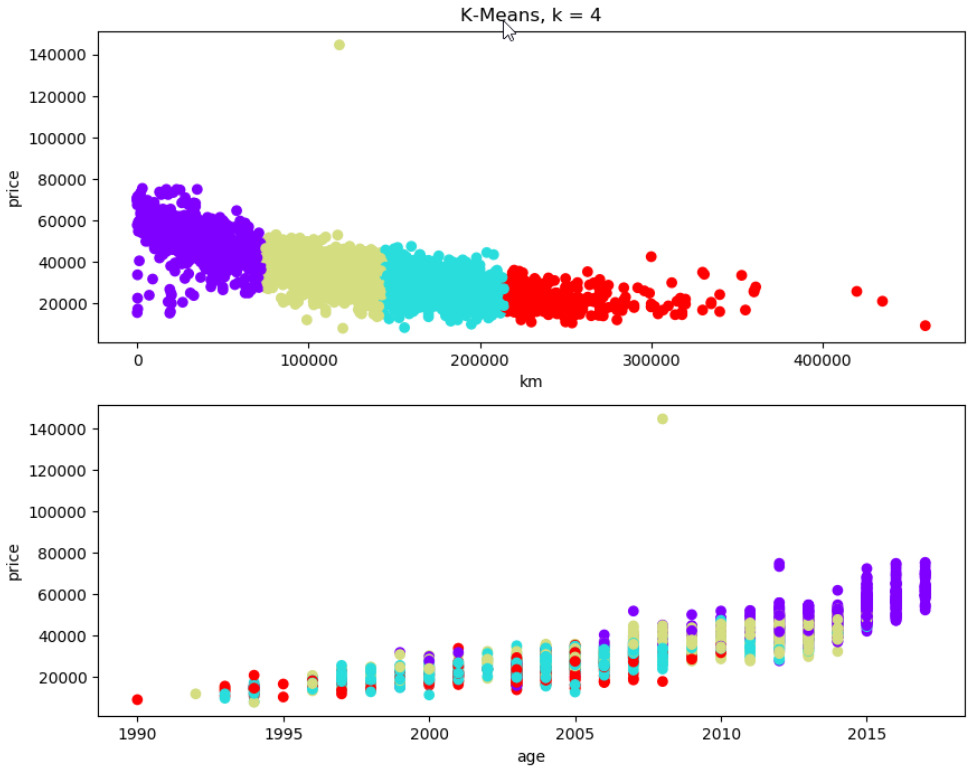
K = 2



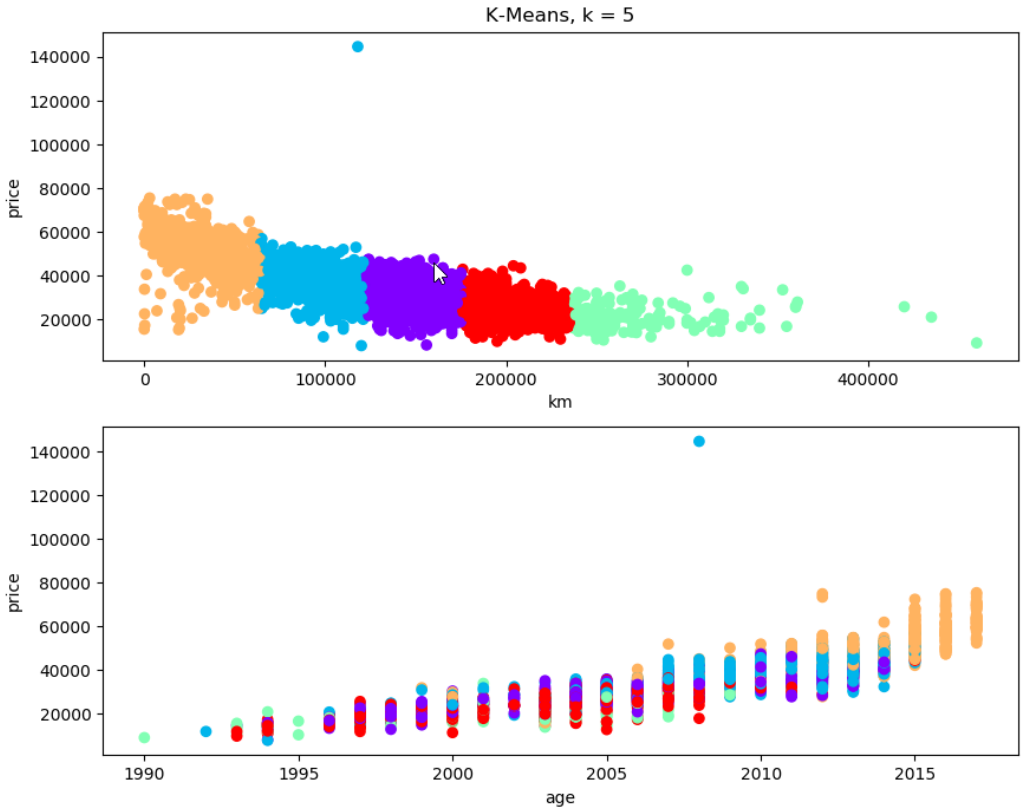
K = 3



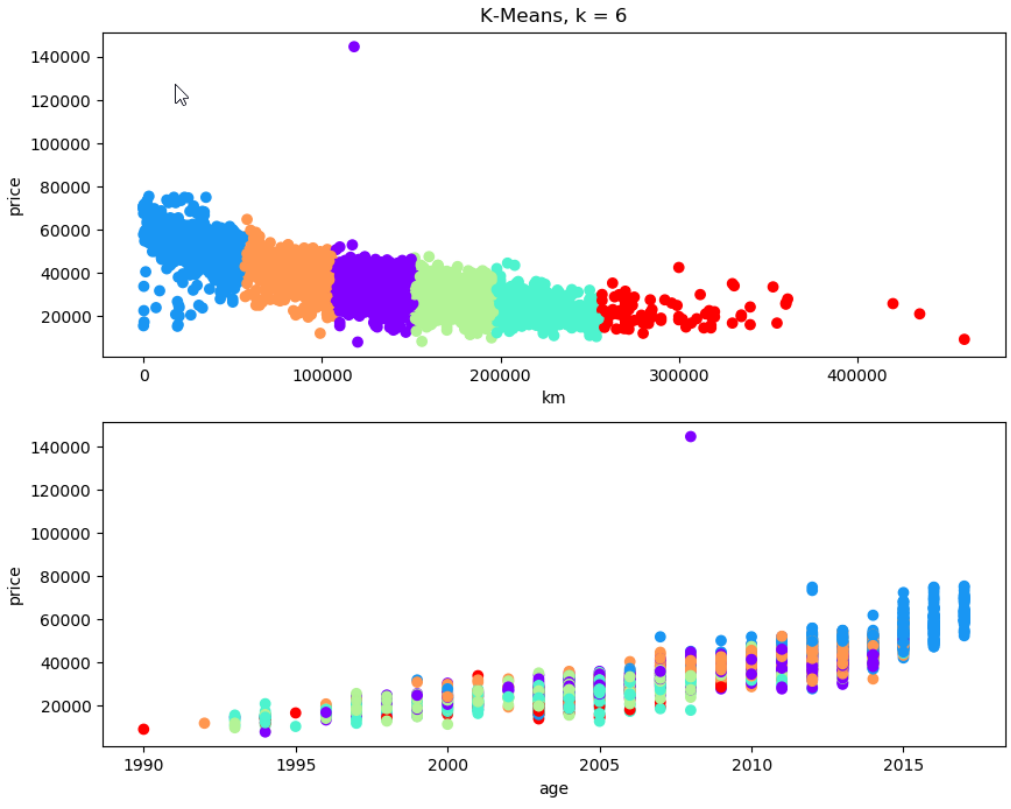
K = 4



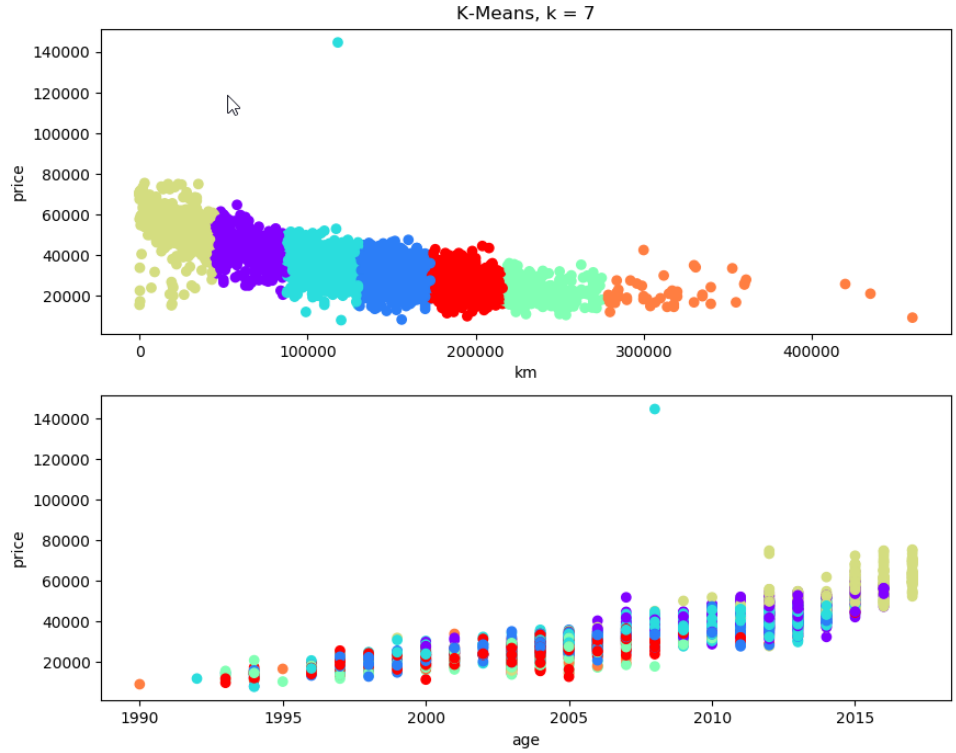
K = 5


K-Means, k = 5
price
km
age
140000
120000
100000
80000
60000
40000
20000
0
100000
200000
300000
400000
1990
1995
2000
2005
2010
2015

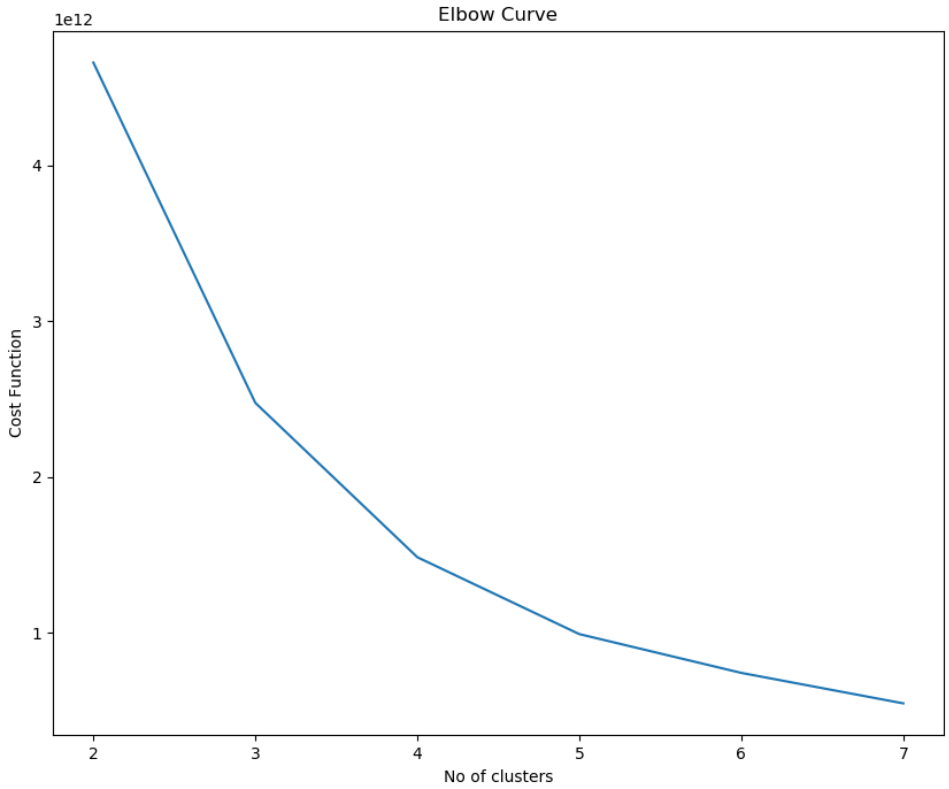
K = 6


K-Means, k = 6
price
km
age
140000
120000
100000
80000
60000
40000
20000
0
100000
200000
300000
400000
1990
1995
2000
2005
2010
2015

K = 7



ELBOW CURVE



```
print(">>>>>>>>>> K-MEAN CLUSTERING MODEL START <<<<<<<<<<")
# K-MEAN CLUSTERING METHOD
# ================================================================
Error =[]
for i in range(2, 8):
  kmeans = KMeans(n_clusters = i, init='k-means++', max_iter=300, n_init=10, random_state=0)
  kmeans.fit(x)
  y_kmeans = kmeans.fit_predict(x)
  Error.append(kmeans.inertia_)

  print("K-Mean Clustering Model Price/km and Price/age curves for k = {}".format(i))
  # Plot Price/km and Price/age curves
# ================================================================
  plt.figure(figsize=(10,8))
  plt.subplot(2,1,1)
  plt.scatter(x[:, 2], y, c = y_kmeans, cmap = 'rainbow')
  plt.xlabel('km')
  plt.ylabel('price')
  plt.title('K-Means, k = {}'.format(i))

  plt.subplot(2,1,2)
  plt.scatter(x[:, 1], y, c = y_kmeans, cmap = 'rainbow')
  plt.xlabel('age')
  plt.ylabel('price')
  plt.show()
# ================================================================

# Plot Elbow Curve
# ================================================================
print('\n')
print('Elbow Curve for k = 2,3,4,5,6,7')
```

```
plt.figure(figsize=(10,8))
plt.plot(range(2, 8), Error)
plt.title('Elbow Curve')
plt.xlabel('No of clusters')
plt.ylabel('Cost Function')
plt.show()
# ======================================================================
# ======================================================================
print('\n')
print(">>>>>>>>>> K-MEAN CLUSTERING MODEL FINISHED <<<<<<<<<<")
```