# КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА



## ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

## Кафедра прикладних інформаційних систем

**Звіт до лабораторної роботи №3**

# з курсу «Big Data»

*Студента 4 курсу*

*Групи ПП-41*

*спеціальності 122*

*«Комп'ютерні науки»*

*ОП«Прикладне програмування»*

*Постоюка Максима*

## Київ – 2023

# Тема

## Методи аналізу та вибору значущих ознак (Features’ Selection Procedures)

# Мета

Метою лабораторної роботи є отримання практичних навичок аналізу та вибору значущих ознак для моделі за допомогою кореляційного аналізу, таблиць сопряжіння, аналізу багатомірні залежності та дихотомії, дисперсійного аналіз – ANOVA, критерій Хі-квадрат тощо.

# Завдання

### Ознайомитись з наданим прикладом використання різних методів відбору значущих ознак (папка Example).

### Завантажити файли з даними у папку проекту з посилання: <https://drive.google.com/file/d/1su22-W8JrRZzm0mea5v8x46YmLh083qp/view?usp=sharing>

### Очистити дані та обробити відсутні дані.

### Зробити EDA по ознаках.

### Проаналізуйте надані дані, використовуючи методи з прикладу та документації, та зберіть результати аналізу у результуючий ранжируваний датафрейм, в якому лівим індексом будуть ознаки, а колонки – результати однофакторного аналізу ознак. Подумайте над системою ранжування такою, яка б врахувала наявність багатьох факторів ранжування (припустимо, що всі вони мають однакову вагу на прийняття вами рішення).

### Проаналізуйте ознаки на взаємозалежність, та побудуйте відповідні heatmap засобами seaborn по кожному з використаних методів дослідження.

### Зберіть висновки у звіт (графіки, висновки текстом у окремому файлі), який потребує належного оформлення, структури тощо.

# Виконання завдання

# Код програми:

import random

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from numpy import set\_printoptions

import numpy as np

import matplotlib as mpl

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import f\_classif

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

from sklearn import preprocessing

from sklearn.decomposition import PCA

import warnings

#from GA import Genetic\_Algorithm # import the python script with GA.

pd.options.mode.chained\_assignment = None  # default='warn'

warnings.simplefilter(action='ignore', category=FutureWarning)

mpl.rcParams["font.size"] = 28

#mpl.rcParams['figure.figsize'] = [18,14]

df = pd.read\_csv('D:/Unik/BigData/Lab3/Data/Automobile\_data.csv', sep = ',')

df2 = pd.read\_csv('D:/Unik/BigData/Lab3/Data/Automobile\_data.csv', sep = ',')

df.loc[df["normalized-losses"]=='?', "normalized-losses"] = "Unknown"

df.loc[df["price"]=='?', "price"] = "Unknown"

df.loc[df["num-of-doors"]=='?', "num-of-doors"] = "Unknown"

df.loc[df["bore"]=='?', "bore"] = "Unknown"

df.loc[df["stroke"]=='?', "stroke"] = "Unknown"

df.loc[df["peak-rpm"]=='?', "peak-rpm"] = "Unknown"

df.loc[df["horsepower"]=='?', "horsepower"] = "Unknown"

indexNormalizedLosses = df[df['normalized-losses'] == 'Unknown'].index

df.drop(indexNormalizedLosses , inplace=True)

indexPrice = df[df['price'] == 'Unknown'].index

df.drop(indexPrice , inplace=True)

indexNumOfDoors = df[df['num-of-doors'] == 'Unknown'].index

df.drop(indexNumOfDoors , inplace=True)

indexBore = df[df['bore'] == 'Unknown'].index

df.drop(indexBore , inplace=True)

indexStroke = df[df['stroke'] == 'Unknown'].index

df.drop(indexStroke , inplace=True)

indexPeakRPM = df[df['peak-rpm'] == 'Unknown'].index

df.drop(indexPeakRPM , inplace=True)

indexHorsepower = df[df['horsepower'] == 'Unknown'].index

df.drop(indexHorsepower , inplace=True)

label\_encoder = preprocessing.LabelEncoder()

dfForEncoder = df

dfForEncoder['make']= label\_encoder.fit\_transform(dfForEncoder['make'])

dfForEncoder['fuel-type']= label\_encoder.fit\_transform(dfForEncoder['fuel-type'])

dfForEncoder['aspiration']= label\_encoder.fit\_transform(dfForEncoder['aspiration'])

dfForEncoder['num-of-doors']= label\_encoder.fit\_transform(dfForEncoder['num-of-doors'])

dfForEncoder['body-style']= label\_encoder.fit\_transform(dfForEncoder['body-style'])

dfForEncoder['drive-wheels']= label\_encoder.fit\_transform(dfForEncoder['drive-wheels'])

dfForEncoder['engine-location']= label\_encoder.fit\_transform(dfForEncoder['engine-location'])

dfForEncoder['engine-type']= label\_encoder.fit\_transform(dfForEncoder['engine-type'])

dfForEncoder['num-of-cylinders']= label\_encoder.fit\_transform(dfForEncoder['num-of-cylinders'])

dfForEncoder['fuel-system']= label\_encoder.fit\_transform(dfForEncoder['fuel-system'])

#Feature selection

array = dfForEncoder.values

X = array[:,0:24]

Y = array[:,24]

Y = Y.astype('float')

# feature extraction

model = ExtraTreesClassifier(n\_estimators=10)

model.fit(X, Y)

print(model.feature\_importances\_)

df\_firstMethod\_Numeric = df[['highway-mpg', 'horsepower', 'stroke']]

df\_firstMethod\_Numeric['stroke'] = df\_firstMethod\_Numeric['stroke'].astype('float')

df\_firstMethod\_Numeric['horsepower'] = df\_firstMethod\_Numeric['horsepower'].astype('float')

df\_firstMethod\_Numeric['highway-mpg'] = df\_firstMethod\_Numeric['highway-mpg'].astype('float')

df\_firstMethod\_String = df2[['engine-type']]

df\_firstMethod\_describe = pd.DataFrame()

df\_firstMethod\_describe['mean'] = df\_firstMethod\_Numeric.mean()

df\_firstMethod\_describe['median'] = df\_firstMethod\_Numeric.median()

df\_firstMethod\_describe['max'] = df\_firstMethod\_Numeric.max()

df\_firstMethod\_describe['min'] = df\_firstMethod\_Numeric.min()

df\_firstMethod\_meanGraph = sns.barplot(x='index', y= 'mean', data=df\_firstMethod\_describe['mean'].reset\_index(name = "mean"))

plt.xlabel('Feature')

df\_firstMethod\_meanGraph.set(title = 'Mean values of the features')

plt.show()

df\_firstMethod\_medianGraph =  sns.barplot(x='index', y= 'median', data=df\_firstMethod\_describe['median'].reset\_index(name = "median"))

plt.xlabel('Feature')

df\_firstMethod\_medianGraph.set(title = 'Median values of the features')

plt.show()

df\_firstMethod\_maxGraph =  sns.barplot(x='index', y= 'max', data=df\_firstMethod\_describe['max'].reset\_index(name = "max"))

plt.xlabel('Feature')

df\_firstMethod\_maxGraph.set(title = 'Max values of the features')

plt.show()

df\_firstMethod\_minGraph =  sns.barplot(x='index', y= 'min', data=df\_firstMethod\_describe['min'].reset\_index(name = "min"))

plt.xlabel('Feature')

df\_firstMethod\_minGraph.set(title = 'Min values of the features')

plt.show()

df\_firstMethod\_enginetype =  sns.barplot(x='count', y= 'engine-type', data=df\_firstMethod\_String.groupby('engine-type').size().reset\_index(name = "count"))

plt.xlabel('count')

df\_firstMethod\_enginetype.set(title = 'Count of each engine-type')

plt.show()

df\_firstMethod\_correlation = sns.heatmap(df[['highway-mpg', 'engine-type', 'stroke', 'horsepower']].corr(), annot=True)

df\_firstMethod\_correlation.set(title = 'Correlation')

plt.show()

#Recursive feature elimination

array = dfForEncoder.values

X = array[:,0:24]

Y = array[:,24]

Y = Y.astype('float')

# feature extraction

model = LogisticRegression(solver='liblinear', max\_iter=200)

rfe = RFE(model,  n\_features\_to\_select=4, step=1)

fit = rfe.fit(X, Y)

print("Num Features: %d" % fit.n\_features\_)

print("Selected Features: %s" % fit.support\_)

print("Feature Ranking: %s" % fit.ranking\_)

df\_secondMethod\_Numeric = df[['stroke', 'compression-ratio']]

df\_secondMethod\_Numeric['stroke'] = df\_secondMethod\_Numeric['stroke'].astype('float')

df\_secondMethod\_Numeric['compression-ratio'] = df\_secondMethod\_Numeric['compression-ratio'].astype('float')

df\_secondMethod\_String = df2[['num-of-cylinders', 'aspiration']]

df\_secondMethod\_describe = pd.DataFrame()

df\_secondMethod\_describe['mean'] = df\_secondMethod\_Numeric.mean()

df\_secondMethod\_describe['median'] = df\_secondMethod\_Numeric.median()

df\_secondMethod\_describe['max'] = df\_secondMethod\_Numeric.max()

df\_secondMethod\_describe['min'] = df\_secondMethod\_Numeric.min()

df\_secondMethod\_meanGraph = sns.barplot(x='index', y= 'mean', data=df\_secondMethod\_describe['mean'].reset\_index(name = "mean"))

plt.xlabel('Feature')

df\_secondMethod\_meanGraph.set(title = 'Mean values of the features')

plt.show()

df\_secondMethod\_medianGraph =  sns.barplot(x='index', y= 'median', data=df\_secondMethod\_describe['median'].reset\_index(name = "median"))

plt.xlabel('Feature')

df\_secondMethod\_medianGraph.set(title = 'Median values of the features')

plt.show()

df\_secondMethod\_maxGraph =  sns.barplot(x='index', y= 'max', data=df\_secondMethod\_describe['max'].reset\_index(name = "max"))

plt.xlabel('Feature')

df\_secondMethod\_maxGraph.set(title = 'Max values of the features')

plt.show()

df\_secondMethod\_minGraph =  sns.barplot(x='index', y= 'min', data=df\_secondMethod\_describe['min'].reset\_index(name = "min"))

plt.xlabel('Feature')

df\_secondMethod\_minGraph.set(title = 'Min values of the features')

plt.show()

df\_secondMethod\_fuelSystem =  sns.barplot(x='count', y= 'aspiration', data=df\_secondMethod\_String.groupby('aspiration').size().reset\_index(name = "count"))

plt.xlabel('count')

df\_secondMethod\_fuelSystem.set(title = 'Count of each aspiration')

plt.show()

df\_secondMethod\_engineType =  sns.barplot(x='count', y= 'num-of-cylinders', data=df\_secondMethod\_String.groupby('num-of-cylinders').size().reset\_index(name = "count"))

plt.xlabel('count')

df\_secondMethod\_engineType.set(title = 'Count of each num-of-cylinders')

plt.show()

df\_secondMethod\_correlation = sns.heatmap(df[['stroke', 'compression-ratio', 'num-of-cylinders', 'aspiration']].corr(), annot=True)

df\_secondMethod\_correlation.set(title = 'Correlation')

plt.show()

#Univariate Selection

array = dfForEncoder.values

X = array[:,0:24]

Y = array[:,24]

Y = Y.astype('float')

# feature extraction

test = SelectKBest(score\_func=f\_classif, k=4)

fit = test.fit(X, Y)

# summarize scores

set\_printoptions(precision=3)

print(fit.scores\_)

features = fit.transform(X)

# summarize selected features

print(features[0:5,:])

df\_thirdMethodNumeric = df[['highway-mpg', 'peak-rpm']]

df\_thirdMethodNumeric['highway-mpg'] = df\_thirdMethodNumeric['highway-mpg'].astype('float')

df\_thirdMethodNumeric['peak-rpm'] = df\_thirdMethodNumeric['peak-rpm'].astype('float')

df\_thirdMethodString = df2[['engine-type', 'fuel-system']]

df\_thirdMethod\_describeNumeric = pd.DataFrame()

df\_thirdMethod\_describeNumeric['mean'] = df\_thirdMethodNumeric.mean()

df\_thirdMethod\_describeNumeric['median'] = df\_thirdMethodNumeric.median()

df\_thirdMethod\_describeNumeric['max'] = df\_thirdMethodNumeric.max()

df\_thirdMethod\_describeNumeric['min'] = df\_thirdMethodNumeric.min()

df\_thirdMethod\_meanGraph = sns.barplot(x='index', y= 'mean', data=df\_thirdMethod\_describeNumeric['mean'].reset\_index(name = "mean"))

plt.xlabel('Feature')

df\_thirdMethod\_meanGraph.set(title = 'Mean values of the features')

plt.show()

df\_thirdMethod\_medianGraph =  sns.barplot(x='index', y= 'median', data=df\_thirdMethod\_describeNumeric['median'].reset\_index(name = "median"))

plt.xlabel('Feature')

df\_thirdMethod\_medianGraph.set(title = 'Median values of the features')

plt.show()

df\_thirdMethod\_maxGraph =  sns.barplot(x='index', y= 'max', data=df\_thirdMethod\_describeNumeric['max'].reset\_index(name = "max"))

plt.xlabel('Feature')

df\_thirdMethod\_maxGraph.set(title = 'Max values of the features')

plt.show()

df\_thirdMethod\_minGraph =  sns.barplot(x='index', y= 'min', data=df\_thirdMethod\_describeNumeric['min'].reset\_index(name = "min"))

plt.xlabel('Feature')

df\_thirdMethod\_minGraph.set(title = 'Min values of the features')

plt.show()

df\_thirdMethod\_fuelSystem =  sns.barplot(x='count', y= 'fuel-system', data=df\_thirdMethodString.groupby('fuel-system').size().reset\_index(name = "count"))

plt.xlabel('count')

df\_thirdMethod\_fuelSystem.set(title = 'Count of each fuel-system')

plt.show()

df\_thirdMethod\_engineType =  sns.barplot(x='count', y= 'engine-type', data=df\_thirdMethodString.groupby('engine-type').size().reset\_index(name = "count"))

plt.xlabel('count')

df\_thirdMethod\_engineType.set(title = 'Count of each engine type')

plt.show()

df\_thirdMethod\_correlation = sns.heatmap(df[['highway-mpg', 'fuel-system', 'engine-type', 'peak-rpm']].corr(), annot=True)

df\_thirdMethod\_correlation.set(title = 'Correlation')

plt.show()

**Висновок**

Отримав практичні навички аналізу та вибору значущих ознак для моделі за допомогою кореляційного аналізу, таблиць сопряжіння, аналізу багатомірних залежностей та дихотомії, дисперсійного аналізу – ANOVA, критерію Хі-квадрат тощо. Різні алгоритми виділяють різні ознаки, які впливають на кінцеву ціну автомобіля, але серед них є ознаки, що були обрані у 2 алгоритмах: витрата бензину на шосе, тип двигуна та хід двигуна. На них варто звернути особливу увагу при виборі автомобілів для продажу. Також була помічена значна кореляція між такими ознаками: кінські сили та витрати бензину на шосе з коефіцієнтом -0.83, система постачання повітря та ступінь стиснення з коефіцієнтом 0.34 та витрата бензину на шосе та система заправки з коефіцієнтом -0.63.