Data Mining - Lab 06 - Feature Scaling and Regression

Nguyễn Đức Vũ Duy - 18110004

```
#Import libraries
# Supress Warnings

import warnings
warnings.filterwarnings('ignore')

# Import the numpy and pandas package
import numpy as np
import pandas as pd

# Data Visualisation
import matplotlib.pyplot as plt
import seaborn as sns

# Import library for label encoder
from sklearn.preprocessing import LabelEncoder

# Import library for scaling
from sklearn.preprocessing import MinMaxScaler

# Import library for spliting training and testing data
from sklearn.model_selection import train_test_split

# Import library of RMSE
from sklearn.metrics import mean_squared_error

# Import libraries for model Building
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
```

Sau khi làm lại 2 bài trên thì áp dụng kết hợp các bước ở trên để thực hiện giải bài toán Regression tốt nhất cho dữ liệu sau “insurance.csv”. Cần dự đoán cột charges.

Trong bài tập “Regression Methods (Customer Churn Analysis)” ta có các bước :

Data Integration

Data Cleansing

EDA

Encoding

Feature Scaling

Feature Selection (Feature Importance)

Model Building

Model Evaluation

Data Integration

```
#Import dataset
path='https://raw.githubusercontent.com/duynguyenhcmus/Repository/main/Hock
df=pd.read_csv(path)
df.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
#Print data information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
```

```
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

```
#Print data describe
df.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

Data cleaning

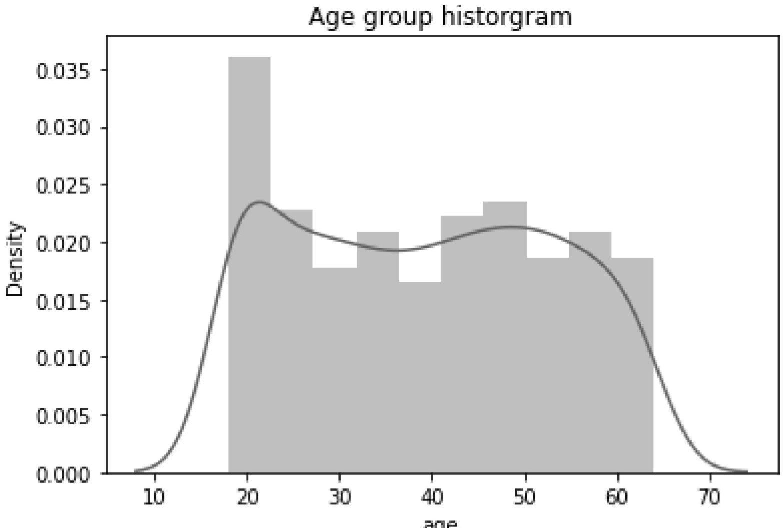
```
#Check for null values
df.isnull().sum()*100/df.shape[0]
#No null values
```

```
age         0.0
sex         0.0
bmi         0.0
children    0.0
smoker      0.0
region      0.0
charges     0.0
dtype: float64
```

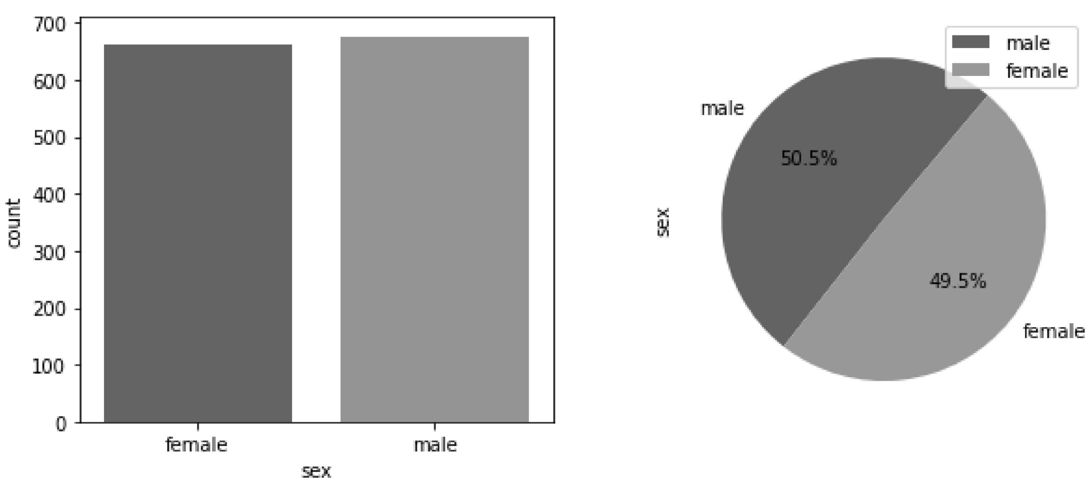
EDA

```
#Age features
fig,ax=plt.subplots()
sns.distplot(a=df.age, bins=10,ax=ax)
plt.title('Age group historgram')
#Not a normal distribution with a majority of people is at the age range of
```

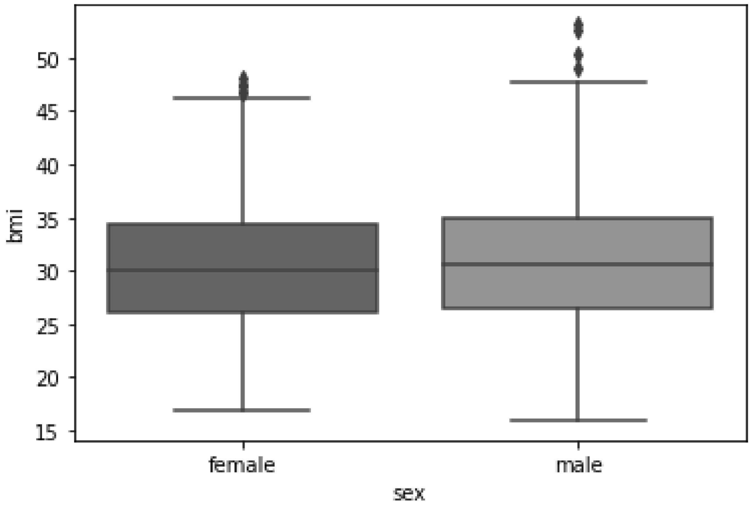
```
Text(0.5, 1.0, 'Age group historgram')
```




```
#Sex features
fig, axs = plt.subplots(1,2, figsize = (10,4))
plt1 = sns.countplot(df['sex'], ax = axs[0])

pie_churn = pd.DataFrame(df['sex'].value_counts())
pie_churn.plot.pie( subplots=True,labels = pie_churn.index.values, autopct=
# Unsquish the pie.
plt.gca().set_aspect('equal')

plt.show()
#It's balance between 2 sex group
```

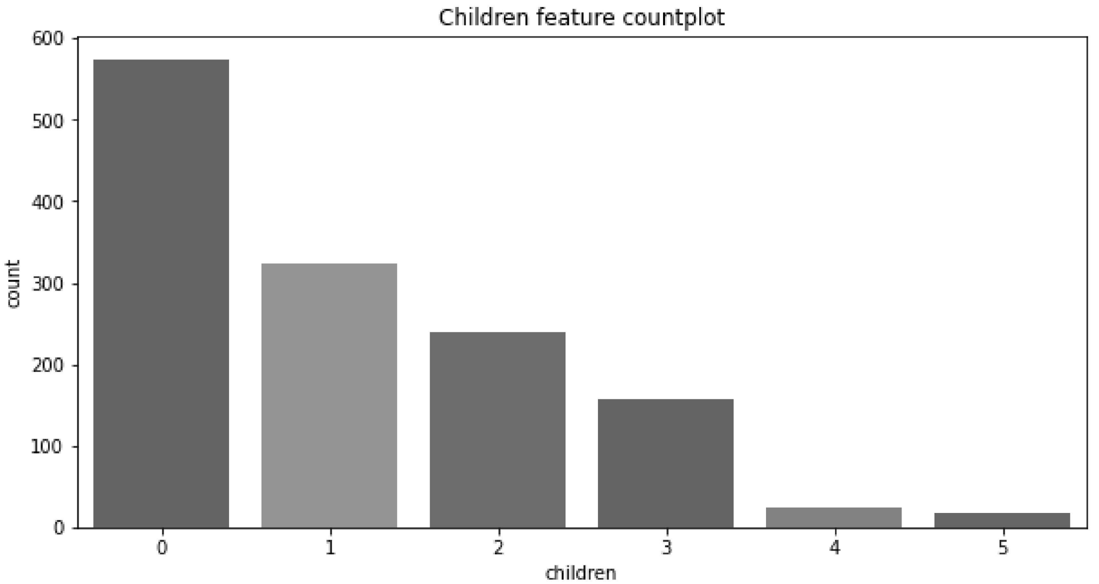



```
#bmi features
sns.boxplot(x = 'sex', y = 'bmi', data = df)
plt.show()
#Male tend to get more outlier than woman. BMI score between 2 sexes is bal
```



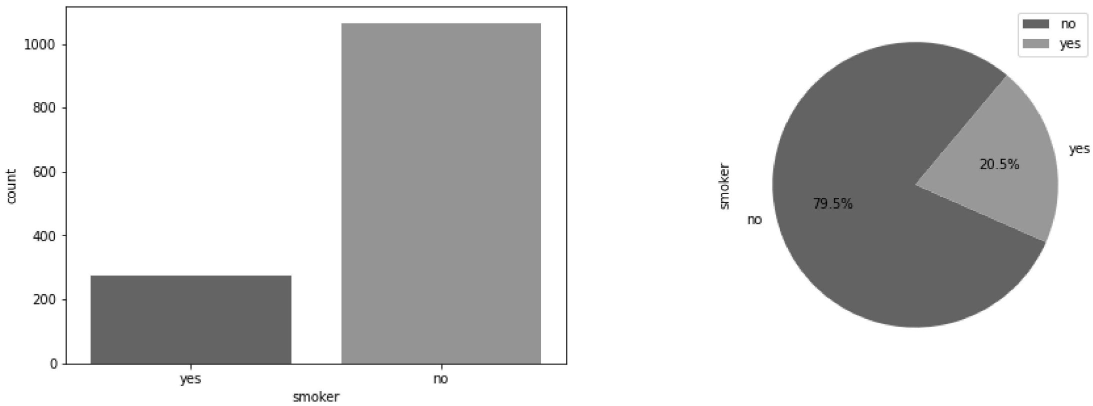

```
#children feature
fig, axs = plt.subplots(figsize = (10,5))
sns.countplot(df['children'], ax = axs)
plt.title('Children feature countplot')
#It's not balance with children at group 4 and 5
```

```
Text(0.5, 1.0, 'Children feature countplot')
```



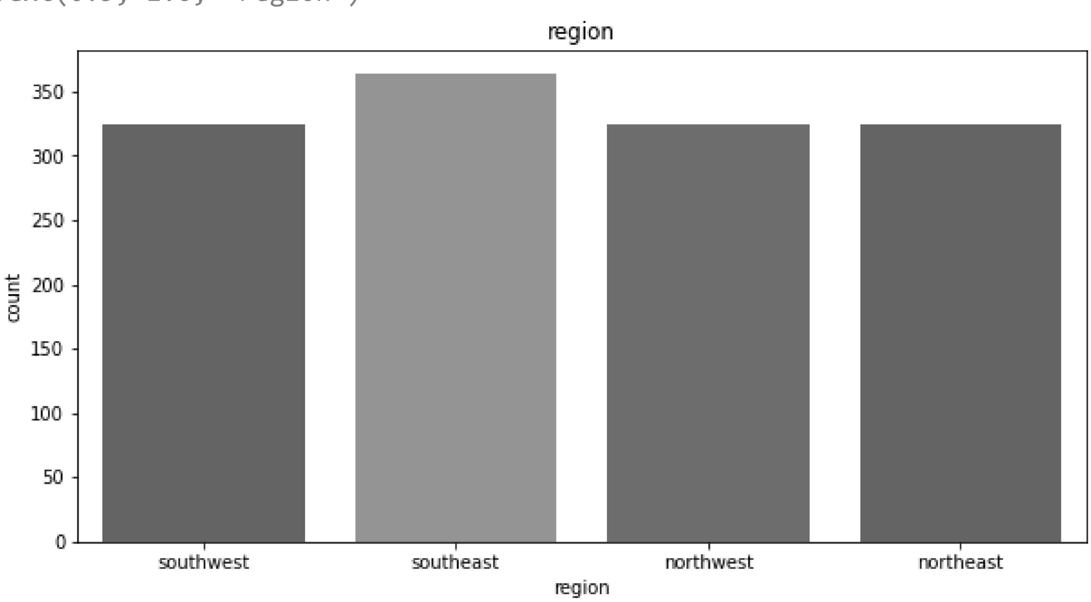

```
#smoking feature
fig, axs = plt.subplots(1,2, figsize = (15,5))
plt1 = sns.countplot(df['smoker'], ax = axs[0])

pie_churn = pd.DataFrame(df['smoker'].value_counts())
pie_churn.plot.pie( subplots=True,labels = pie_churn.index.values, autopct=
# Unsquish the pie.
plt.gca().set_aspect('equal')
```

```
plt.show()
#A majority of people is non-smoker
```



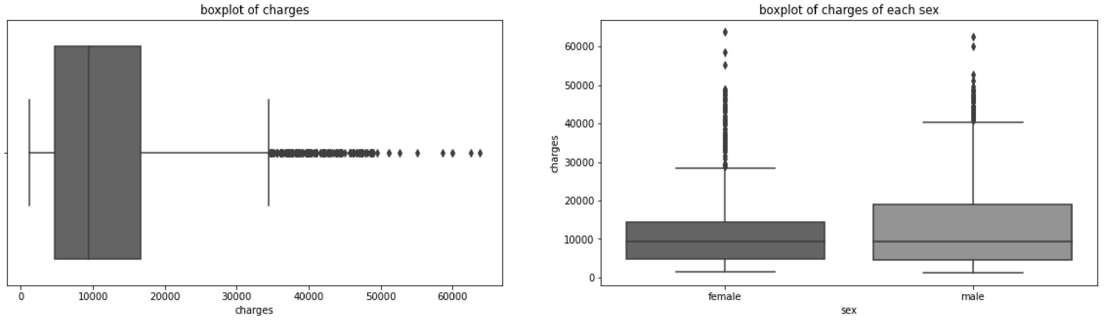

```
#region feature
fig, axs = plt.subplots(figsize = (10,5))
sns.countplot(df['region'], ax = axs)
plt.title('region')
#It's quite balance between the number of different groups
```

```
Text(0.5, 1.0, 'region')
```




```
#Charge as target
```

```
fig, axs = plt.subplots(1,2, figsize = (20,5))
plt1 = sns.boxplot(df['charges'], ax = axs[0])
plt1.set_title('boxplot of charges')
plt2 = sns.boxplot(data=df, x='sex',y='charges',ax=axs[1])
plt2.set_title('boxplot of charges of each sex')

plt.show()
#Both sex and each sex have a lot of outliers
```




```
#Removing outliers
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print('> Shape of data before handling outlier values: ', df.shape)
df = df[~((df<(Q1-1.5*IQR))|(df>(Q3+1.5*IQR))).any(axis=1)]
print('> Shape of data after handling outlier values: ', df.shape)
```

```
> Shape of data before handling outlier values:  (1338, 7)
> Shape of data after handling outlier values:  (1193, 7)
```

Encoding

```
#Using label encoding from sklearn
label = LabelEncoder()
data_colums = df.dtypes.pipe(lambda X: X[X=='object']).index
for col in data_colums:
    df[col] = label.fit_transform(df[col])

```

```
df.info()
#It's all numeric data now
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1193 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1193 non-null   int64  
 1   sex       1193 non-null   int64  
 2   bmi       1193 non-null   float64
 3   children  1193 non-null   int64  
 4   smoker    1193 non-null   int64  
 5   region    1193 non-null   int64  
 6   charges   1193 non-null   float64
dtypes: float64(2), int64(5)
memory usage: 74.6 KB
```

Feature scaling

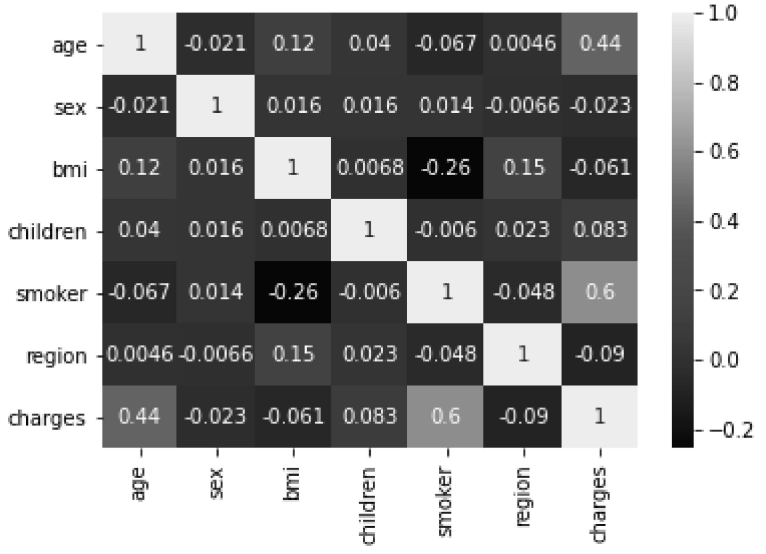
```
#Train test split for feature scaling
X=df.iloc[:,:-1].copy()
y=df.iloc[:,-1].copy()
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,rand
```

```
# copy of datasets
X_train_norm = X_train.copy()
X_test_norm = X_test.copy()

# numerical features
num_cols = ['age','bmi']
# apply standardization on numerical features
for i in num_cols:
    # fit scaler on training data
    norm = MinMaxScaler().fit(X_train_norm[[i]])
    # transform the training data column
    X_train_norm[i] = norm.transform(X_train_norm[[i]])
    # transform the testing data column
    X_test_norm[i] = norm.transform(X_test_norm[[i]])
```

Feature Selection

```
# Let's see the correlation matrix
plt.figure()
sns.heatmap(df.corr(),annot = True)
```

```
plt.show()
#The correlation between features is quite low. So there is no implication
#However, sex, bmi, children and region have very low correlation with the
```




```
#Remove uncorrelated features
df=df.drop(['sex','bmi','children','region'],axis=1)
df.head()
```

	age	smoker	charges
0	19	1	16884.92400
1	18	0	1725.55230
2	28	0	4449.46200
3	33	0	21984.47061
4	32	0	3866.85520

Hãy sử dụng các thuật toán Regression : Linear, Ridge, Lasso, Decision Tree, K Neighbour, Random Forest Regression và so sánh RMSE.

Model building

```
#Build Model
rmse=[]

```

```
#Linear Regression
linear_regression=LinearRegression().fit(X_train, y_train)
y_linear_regression=linear_regression.predict(X_test)
rmse.append(mean_squared_error(y_linear_regression,y_test,squared=False))

#Ridge Regression
ridge_regression=Ridge(alpha=1).fit(X_train,y_train)
y_ridge_regression=ridge_regression.predict(X_test)
rmse.append(mean_squared_error(y_ridge_regression,y_test,squared=False))

#Lasso Regression
lasso_regression=Lasso(alpha=1).fit(X_train,y_train)
y_lasso_regression=lasso_regression.predict(X_test)
rmse.append(mean_squared_error(y_lasso_regression,y_test,squared=False))

#Decision Tree Regression
decision_tree=DecisionTreeRegressor(max_depth=5, random_state=0).fit(X_trai
y_decision_tree=decision_tree.predict(X_test)
rmse.append(mean_squared_error(y_decision_tree,y_test,squared=False))

#K Neighbors Regression
k_neighbors=KNeighborsRegressor(n_neighbors=10).fit(X_train,y_train)
y_k_neighbors=k_neighbors.predict(X_test)
rmse.append(mean_squared_error(y_k_neighbors,y_test,squared=False))

#Random forest regression
random_forest=RandomForestRegressor(n_estimators=50).fit(X_train,y_train)
y_random_forest=random_forest.predict(X_test)
rmse.append(mean_squared_error(y_random_forest,y_test,squared=False))

```

Model evaluation

```
# visualizing the result
df_dt = pd.DataFrame({'RMSE':rmse},index=['Linear regression','Ridge Regres
df_dt


```

	RMSE
Linear regression	4626.282460
Ridge Regression	4624.517125
Lasso Regression	4625.986412
Decision Tree Regression	4542.606272
K Neighbors Regression	5998.767660
Random Forest Regression	4879.058564

✓ 0s completed at 2:37 PM