Machine Learning Project

By: Hatem Abu sadaa , Mojeeb al Rahmaan Hasan, Mohammed Nimran, Malek Bazbaz,Mohammed abd al-razzaq

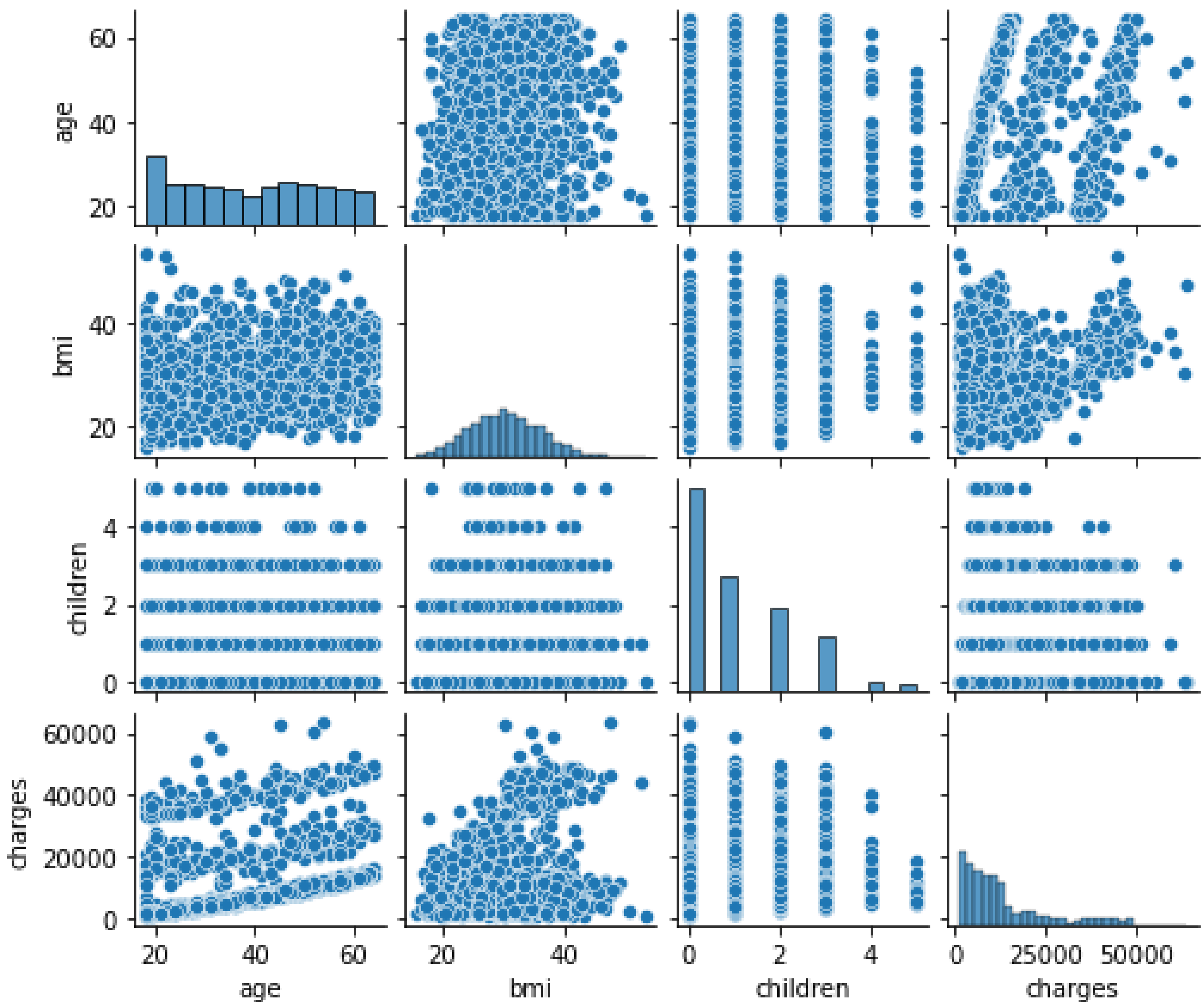
Assignment 1: Regression

We will be discussing the process that we went through to make a Liner regression model, from getting to know the data, preprocessing the data, visualization , and finally building a liner regression model.

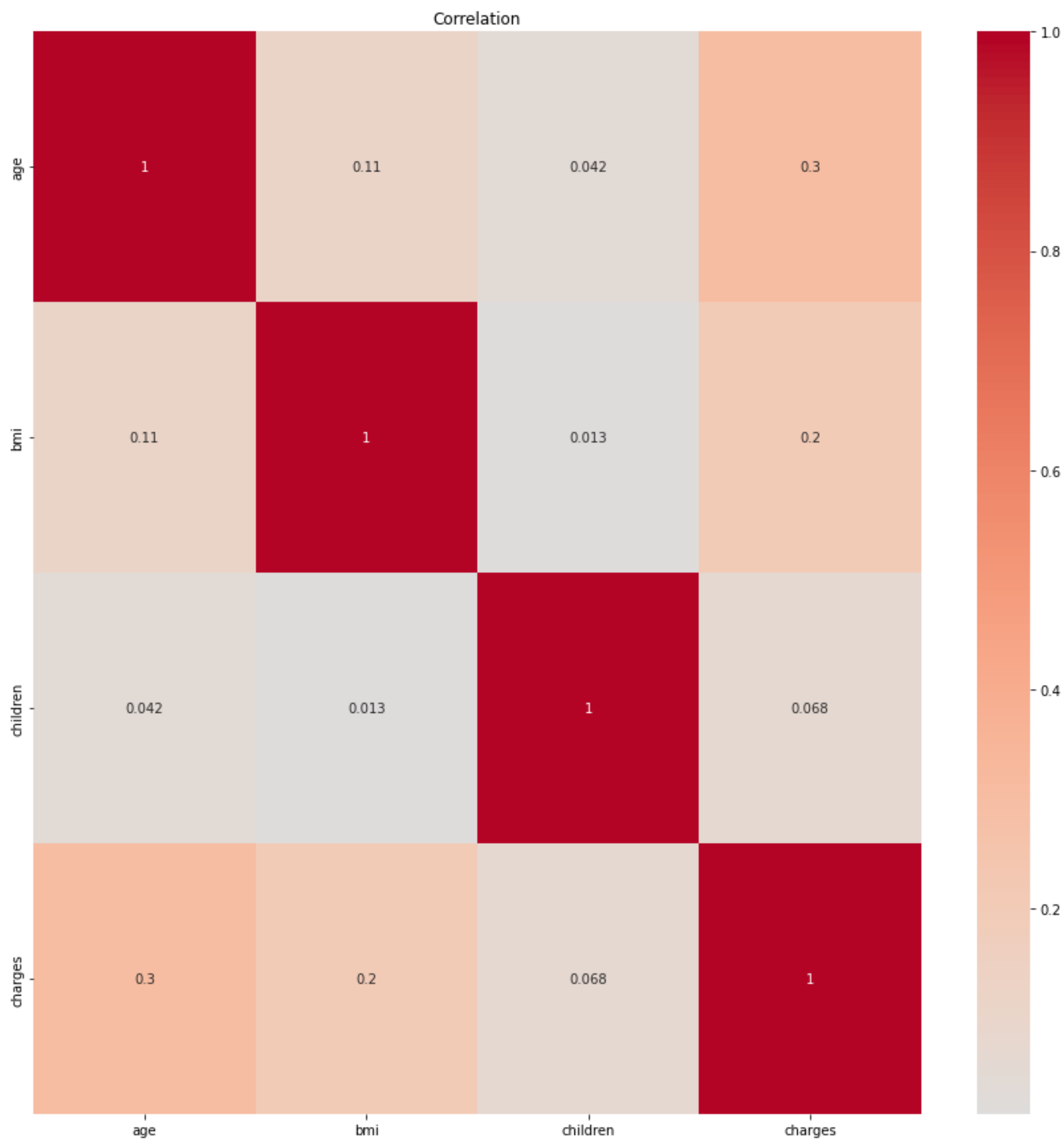
First we checked the data to see if it had some null values, thankfully the data we are with doesn’t have any

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
   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
```

Scatterplot matrix:



Correlation matrix:

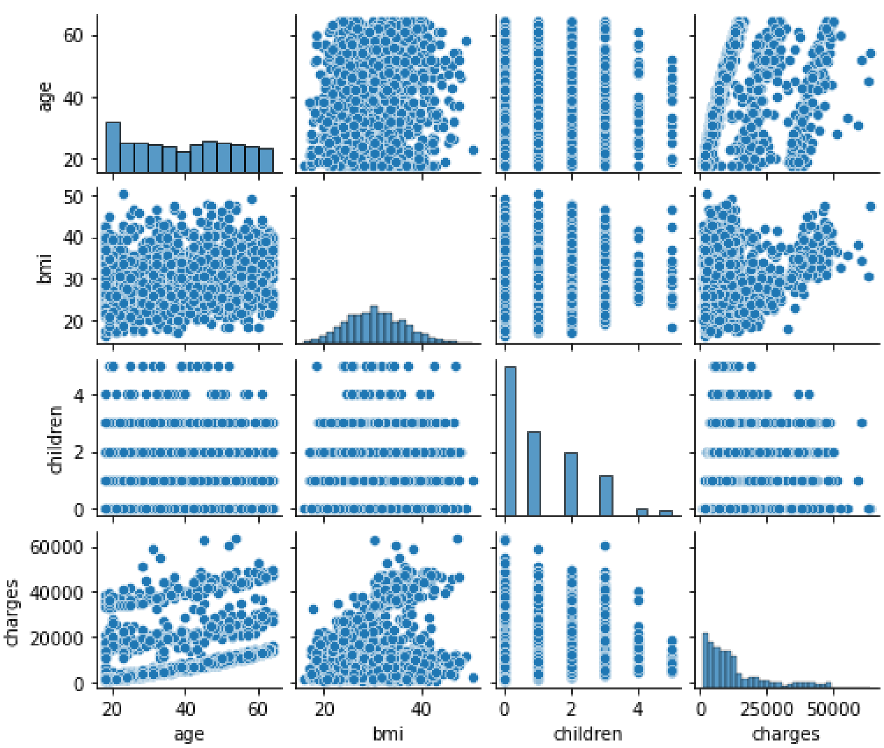


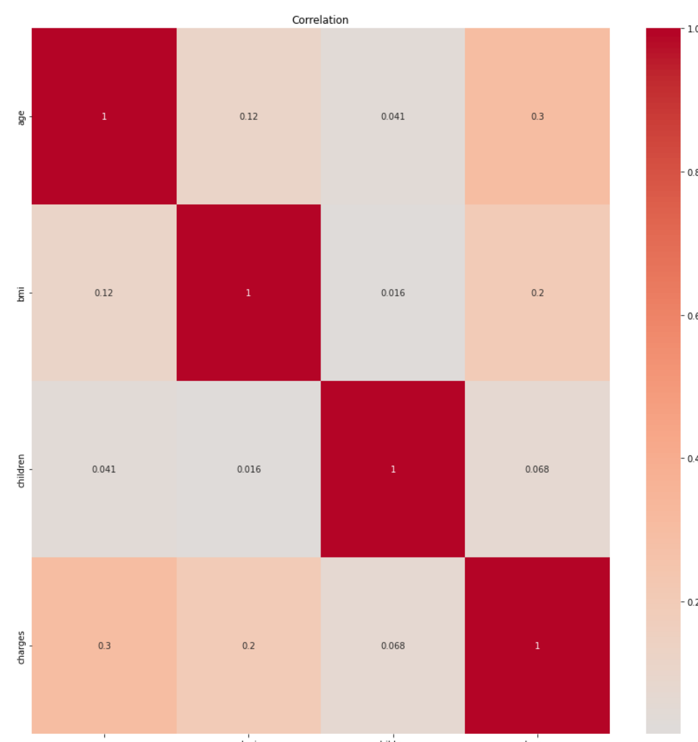
We noticed some outliers in ‘bmi’ and ‘age’ So we decided to get rid of the outliers using interquartile Range (IQR) method.

interquartile Range (IQR) method:

```
def remove_outliers(data, feature_names, iqr_multiplier=1.5):

    data_no_outliers = data.copy()

    for feature in feature_names:
        Q1 = data_no_outliers[feature].quantile(0.25)
        Q3 = data_no_outliers[feature].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - iqr_multiplier * IQR
        upper_bound = Q3 + iqr_multiplier * IQR

        data_no_outliers = data_no_outliers[
            (data_no_outliers[feature] >= lower_bound) &
            (data_no_outliers[feature] <= upper_bound)
        ]

    return data_no_outliers
```

Data Visualization After Removing The Outliers :





Liner Regression models doesn’t accept categorical data so we turned categorical features into numeric.

We used for that two encoding techniques one-hot encoding for ‘region’ feature because its not an ordinal feature and not binary.

```
# Convert categorical data to numerical using one-hot encoding
encoder = OneHotEncoder(sparse=False)
categorical_features = ['region']
encoded_categories = encoder.fit_transform( insurance_no_outliers[categorical_features])
encoded_df = pd.concat([
    insurance_no_outliers.drop(columns=categorical_features),
    pd.DataFrame(encoded_categories, columns=encoder.get_feature_names(categorical_features))
], axis=1)
```

And label encoding for ‘sex’ and ‘smoker’ because they are both binary features.

```
# Convert other categorical features using label encoding
encoder1 = LabelEncoder()
categorical_features_label = ['sex', 'smoker']
encoded_df[categorical_features_label] = encoded_df[categorical_features_label].apply(encoder1.fit_transform)
```

After the encoding process some null values appeared so we decided to get rid of them.

```
encoded_df=encoded_df.dropna()
```

The final step of preprocessing was normalizing the data.

```
# Scale the data
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

We used Backward selection to know witch features are more important for building the model

```
# Backward Elimination
n_features_to_select = x_train.shape[1]  # Start with all features

for i in range(n_features_to_select, 1, -1):
    rfe = RFE(estimator=reg, n_features_to_select=i)
    x_train_selected = rfe.fit_transform(x_train_scaled, y_train)
    x_test_selected = rfe.transform(x_test_scaled)

    # Get the indices of selected features
    selected_indices = np.where(rfe.support_)[0]
    selected_features = feature_names[selected_indices]

    # Train the model
    reg.fit(x_train_selected, y_train)
    y_pred_train = reg.predict(x_train_selected)
    y_pred_test = reg.predict(x_test_selected)

    # Calculate MSE
    mse_test = np.sqrt(mean_squared_error(y_test, y_pred_test))

    print(f'With {i} features:')
    print(f'Selected features: {selected_features}')
    print(f'Test score (R2): {reg.score(x_test_selected, y_test)}')
    print(f'MSE on test: {mse_test}')
    print("-----------------------------------------------------------------")
```

The results :

```
With 9 features:
Selected features: Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region_northeast',
       'region_northwest', 'region_southeast', 'region_southwest'],
      dtype='object')
Test score (R2): 0.8576122019863063
MSE on test: 5197.320167345304
```

```
With 8 features:
Selected features: Index(['age', 'bmi', 'children', 'smoker', 'region_northeast',
       'region_northwest', 'region_southeast', 'region_southwest'],
      dtype='object')
Test score (R2): 0.8573552745116694
MSE on test: 5202.00712974923
```

```
With 7 features:
Selected features: Index(['age', 'bmi', 'children', 'smoker', 'region_northeast',
       'region_southeast', 'region_southwest'],
      dtype='object')
Test score (R2): 0.8573552745116694
MSE on test: 5202.00712974923
```

```
With 6 features:
Selected features: Index(['age', 'bmi', 'children', 'smoker', 'region_southeast',
       'region_southwest'],
      dtype='object')
Test score (R2): 0.8577155534589305
MSE on test: 5195.433600490848
```

```
With 5 features:
Selected features: Index(['age', 'bmi', 'children', 'smoker', 'region_southwest'],
dtype='object')
Test score (R2): 0.8608606671392711
MSE on test: 5137.691733878679
```

```
With 4 features:
Selected features: Index(['age', 'bmi', 'children', 'smoker'], dtype='object')
Test score (R2): 0.8587124734705125
MSE on test: 5177.2006304325305
```

```
With 3 features:
Selected features: Index(['age', 'bmi', 'smoker'], dtype='object')
Test score (R2): 0.8562279192380927
MSE on test: 5222.523025301825
```

```
With 2 features:
Selected features: Index(['age', 'smoker'], dtype='object')
Test score (R2): 0.8197734886301469
MSE on test: 5847.260230364019
```

```
With 1 features:
Selected features: Index(['smoker'], dtype='object')
Test score (R2): 0.7695353146428062
MSE on test: 6612.189288885871
```

Based on the presented results we concluded that the best model was the mode with the 5 features (age,bmi,children,smoker,region_southwest)

Assignment 2: classification

In this assignment we are going to work with a data set called "Celiac disease.csv" witch is a data we collected via a survey .

Features of the data:

Timestamp

gender

age

Medical diagnosis

relatives_diagnosed :if the person have relatives diagnosed with celiac disease{'yes','no'}

numper_relatives_diagnosed : number of relatives diagnosed with celiac disease(0-5)

type 1 diabetes

anemia

unwanted weight loss

bloating/gas

abdominal pain

vomiting /nausea

diarrhea

constipation

fatigue/stress

itchy, blistery skin rash

lactose intolerance

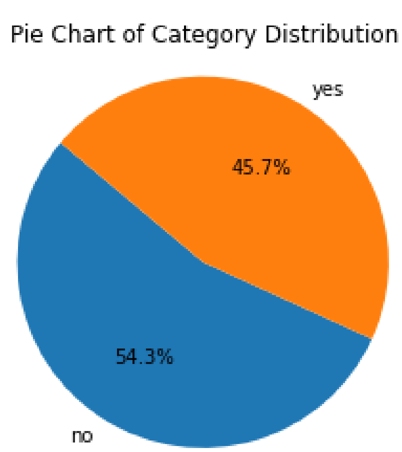
weak bones

Frequent headaches

When analyzing the data we discovered that the feature ‘age’ and the first row of the data contained a lot of noisy data and null values so we decided to get rid of them.

```
[5 rows x 19 columns]
Timestamp                      0
gender                         1
age                           20
Medical diagnosis              0
relatives_diagnosed            1
numper_relatives_diagnosed     1
type 1 diabetes                1
anemia                         1
unwanted weight loss           0
bloating/gas                   1
abdominal pain                 1
vomiting /nausea               1
diarrhea                       1
constipation                   1
fatigue/stress                 1
itchy, blistery skin rash      1
lactose intolerance            1
weak bones                     1
Frequent headaches             1
dtype: int64
```

After removing we wanted to check if the data was unbalanced using pie chart, thankfully the data was balanced enough .



After that each one in the team built a different classification models each came with different results:

Bagging:

```
============================================
Cross Vaildation ==>(Bagging classifier )
Cross-Validation Average Accuracy: 0.7088888888888889
Confusion Matrix:
[[182  63]
 [ 68 137]]
ROC Curve AUC: 0.7781383773021404
============================================
```

```
============================================
Hold out ==> (Bagging Classifier)
Ensemble Classifier Accuracy: 0.6888888888888889
Error Rate: 0.3111111111111111
Confusion Matrix:
[[21  5]
 [ 9 10]]
ROC Curve AUC: 0.7722672064777327
============================================
```

Decision tree:

```
Cross Validation
Mean Cross-Validation Accuracy: 0.7481481481481481
Confusion Matrix:
[[19  7]
 [ 9 10]]
AUC Score: 0.5819838056680162
```

```
Holdout
Accuracy: 0.6444444444444445
Confusion Matrix:
[[19  7]
 [ 9 10]]
AUC: 0.6285425101214575
```

KNN:

```
Mean Accuracy (10-Fold Cross-Validation): 0.70666666666
66667
Confusion Matrix:
[[214  31]
 [ 96 109]]
ROC AUC score(cross validation)    0.7679741164758587
```

```
Accuracy (Hold-out): 0.7333333333333333
Confusion Matrix:
[[25  1]
 [11  8]]
ROC AUC score(hold out)    0.7904858299595142
```

naïve bayes:

```
Cross Validation
Mean AUC: 0.78
================================================================
Confusion Matrix:
[[166  79]
 [ 42 164]]
================================================================
Accuracy: 0.73
================================================================
```

```
Hold-Out
Mean AUC: 0.81
================================================================
Confusion Matrix:
[[30 20]
 [ 4 37]]
================================================================
Accuracy: 0.74
================================================================
```

SVC:

```
cross validation
Mean AUC: 0.83
==============================
Confusion Matrix:
[[200  45]
 [ 72 134]]
==============================
Accuracy: 0.74
```

```
hold out
AUC: 0.83
==============================
confusion_matrix
[[23  3]
 [ 9 10]]
==============================
Accuracy: 0.73
```

After analyzing the results with the team we concluded that the best two models were SVC using cross validation and naïve bayes classifier using hold out method .