A logo with a yellow and green and blue text

Description automatically generated with medium confidence **Cairo University Faculty of Computers and Artificial Intelligence**



**Machine Learning**

**Assignment1**

**Project Team**

|  |  |
| --- | --- |
| **ID** | **Name** |
| 20210382 | Maryam Abd El-Hamid Ibrahim |
| 20210072 | Alaa Hossam Mohammed |
| 20210348 | Mohamed Fathy Sayed |
| 20210229 | Abdel-Rahman Mohamed Ahmed |
| 20210388 | Mostafa Ahmed Mohamed |

**b) Perform analysis on the dataset to:**

* check whether there are missing values
* check whether numeric features have the same scale
* visualize a pair plot in which diagonal subplots are histograms
* visualize a correlation heatmap between numeric columns

**c) Preprocess the data such that:**

* the features and targets are separated
* categorical features and targets are encoded
* the data is shuffled and split into training and testing sets
* numeric features are scaled

**d) Implement linear regression using gradient descent from scratch to predict the CO2 emission amount.**

* Based on the correlation heatmap, select two features to be the independent variables of your model. Those two features should have a strong relationship with the target but not a strong relationship with each other (i.e. they should not be redundant).
* Calculate the cost in every iteration and illustrate (with a plot) how the error of the hypothesis function improves with every iteration of gradient descent.
* Evaluate the model on the test set using Scikit-learn’s R2 score.

**e) Fit a logistic regression model to the data to predict the emission class.**

* Use the two features that you previously used to predict the CO2 emission amount.
* **The Code:**

x\_test = xTest[['Fuel Consumption Comb (L/100 km)', 'Engine Size(L)']].to\_numpy()

* we choose ‘Fuel Consumption Comb (L/100 km)', 'Engine Size(L)' because they have strong relation between CO2 emission amounts (Fuel Consumption Comb (L/100 km)' with percentage 92%, 'Engine Size(L) with percentage 85%)
* The logistic regression model should be a stochastic gradient descent classifier.
* **The Code:**

def run\_log(self):  
 lg = SGDClassifier(max\_iter=500, loss='log\_loss')  
 lg.fit(self.x\_train,self.y\_train)  
 y\_pred = lg.predict(self.x\_test)  
 return y\_pred

In Main  
  
plt.figure(figsize=(8, 6))  
sns.heatmap(confusion\_matrix(y2\_test, y\_pred, labels = [0,1,2,3]), annot=True, fmt='d', cmap='Blues',  
 xticklabels=['HIGH', 'LOW', 'MODERATE', 'VERY LOW'], yticklabels=['HIGH', 'LOW', 'MODERATE', 'VERY LOW'])  
  
plt.xlabel('Predicted' ,labelpad=20)  
plt.ylabel('True Label', labelpad=20)  
plt.title('Confusion Matrix')  
plt.show()

**A screenshot of a computer

Description automatically generated**

* Calculate the accuracy of the model using the test set

def calc\_accuracy(self, y\_pred):  
 accuracy = accuracy\_score(y\_pred, self.y\_test)  
 return accuracy

In Main

accuracy = log\_obj.calc\_accuracy(y\_pred)  
print(f'Accuracy: {accuracy\*100:.2f}')

A screenshot of a computer program

Description automatically generated