

Aprendizaje Automático Práctica 1

Uso de ChatGPT



ChatGPT se utilizó en ciertas áreas de la práctica. En el siguiente documento, expondremos sus casos de uso específicos. Es importante destacar que los prompts y respuestas se redujeron para permitir más casos de uso.

- **Prompt:** I need a definitive way to figure out if the problem is a classification or regression problem.
- **Response:** Given the nature of your problem, it is advisable to use a threshold value. The choice of threshold for distinguishing between a regression and a classification problem based on the number of unique values in the response variable depends on the context and scale of your data. The threshold is not directly related to the number of features or instances but rather to how granular the data is and the nature of the response variable.

-
- **Prompt:** What are some things we can measure for further analysis on our data?
 - **Response:** Correlation Analysis, Distribution Analysis, Outliers Detection:

-
- **Prompt:** Given the statistics shown below:

```
Minimum energy output: 0.01
Maximum energy output: 2640.18
Average energy output: 675.0776631578948
Standard deviation of energy output: 641.9229390122447
```

What is considered a good and bad RMSE for our data? Give me the result in ranges.

- **Response:**
Good RMSE: Less than 320 units.
Acceptable RMSE: 320 to 641 units.
Poor RMSE: Greater than 641 units.

En muchos casos, ChatGPT se utilizó para crear las pipelines y estructuras para contener los modelos. Cuando eran extensas, simplemente pedíamos los modelos y sus requisitos y ChatGPT nos daba el código. Incluiremos dos ejemplos:

- **Prompt:** Give me a dictionary with a pipeline of KNN, regression trees, linear regression normal and lasso, and SVM. Use random state 438770, do not change hyperparameter values.

- **Response:**

```
models = {  
'KNN': make_pipeline(StandardScaler(), KNeighborsRegressor()),  
  
'RegressionTree': make_pipeline(StandardScaler(),  
DecisionTreeRegressor(random_state=438770)),  
  
'LinearRegressionNormal': make_pipeline(StandardScaler(),  
LinearRegression()),  
  
'LinearRegressionLasso': make_pipeline(StandardScaler(),  
Lasso(alpha=1.0, max_iter=10000, random_state=438770)),  
  
'SVM': make_pipeline(StandardScaler(), SVR())  
}
```

The following prompt was also asked for the other models.

- **Prompt:** Give me the hyperparameters of KNN

- **Response:**

```
knn_params = { 'knn__n_neighbors': range(1, 31), 'knn__weights':  
['uniform', 'distance'], 'knn__algorithm': ['auto', 'ball_tree',  
'kd_tree', 'brute']}
```

-
- **Prompt:** Should we use grid search or random search.

- **Response:** Given the nature of your data, it is advisable to use GridSearchCV.
-

- **Prompt:** Give me a dummy regressor to compare the results of the previous models with the dummy regressor.
 - **Response:** Gave code block for a dummy regressor.
-

- **Prompt:** Previously, we compared the RMSE of using different scalars with KNN. Standard Scalar had the lowest RMSE but the difference compared to Robust Scalar is very small. Which scalar should we use?
 - **Response:** Given that the difference between the two mentioned Scalars is small, it is not incorrect to use either one. The answer depends on the nature of your data. Since you have already adapted the solution to the Standard Scalar, it is advisable to continue using it.
-

- **Prompt:** SVM saw a great improvement in the outer evaluation. KNN still has the lowest RMSE for inner but SVM has a lower one for outer. Which should I use for the final model in order to make predictions on the second dataset?
 - **Response:** Seeing little improvement in KNN could indicate that the data was overfitted. Without having access to your data, no definitive conclusion can be made. For that reason, given the minor differences in their RMSE, you may use either one to make predictions on your data.
-

Este problema fue resuelto por el profe.

- **Prompt:** What can I do to improve the model.
- **Response:**

- Re-evaluate Feature Scaling
- Check Data Leakage
- Analyze the Distribution of Predictions
- Evaluate Model Assumptions
- Investigate Overfitting or Underfitting
- Feature Engineering
- Outliers in the Data

- Model Complexity
- Seek Domain Insights
- Experiment with Different Models

-
- **Prompt:** How do I transform the problem into a classification one?
 - **Response:**

Step 1: Define High and Low Energy Outputs

First, you'll need to determine what constitutes "low" and "high" energy outputs. This can be based on the distribution of energy output values in your dataset. For simplicity, let's use the median as the threshold, which divides your dataset into two equally sized groups. However, since the task specifies using the third quartile, we'll use that to distinguish between low and high values.

Step 2: Separate Predictions into Low and High Groups

Once you've identified the threshold (the third quartile), classify each actual energy output as either low or high and do the same for the predictions. This allows you to compare the model's performance on low versus high energy outputs.

Step 3: Evaluate Model Performance on Low vs. High Energy Outputs

Evaluate how well your model predicts low and high energy outputs by comparing the actual and predicted classifications. You might use metrics like RMSE separately for low and high outputs to see if there's a significant difference in performance.