Aprendizagem Automática 23/24

Third Home Assignment

Grupo 33

Authors: Guilherme Cepeda – 62931 horas: 51

Guilherme Rosário – 62543 horas: 4

Marco Viana – 62550 horas: 4

The code, report and predictions are available online through a GitHub repository here.

1 - Introduction

The data provided consists of a .pickle file with 7337 entries, each containing data regarding the molecular activity of molecules over the Dopamine2 Receptors. Each column has information about the molecular activity, due to the different kind of data we divided it into two parts the D columns and the FP columns.

The assignment consists of providing the best possible regression models regarding the target variable in "y_train", using methods and models covered in class.

2 - Exploratory Data Analysis (EDA)

Our focus in EDA was to understand the data and its structure. We looked at the data types, missing values, and other characteristics of the data. Discovered and visualized the data to gain insights.

We started our Exploratory Data Analysis by checking if there was any duplicated data and any missing values in our dataset. After obtaining the statistical information of the "X_train" dataset and the "y_train" dataset, we then removed all the duplicated values after joining the X_train and y_train into one dataset as we can't directly remove the duplicates in the train set because the data was already divided. This is a data processing task but done in EDA to improve our data analysis, without it our EDA would be less meaningful because we wouldn't be able to analyze the correct number of rows when we divide our training set into 2 parts:

- A dataset for the D columns present in the training set only ("**df_X_train_D**").
- A dataset for the FP columns present in the training set only ("**df_X_train_FP**").

Before plotting the histograms, we noticed that the dataset **df_X_train_FP** had only 0s and 1s so it wouldn't be meaningful to plot the histograms for this dataset.

After plotting the histograms for each column of the dataset **df_X_train_D**, we can conclude that the columns ['D12', 'D13','D14','D15','D16','D17','D18','D19','D20','D21','D23'] are numeric categories and should be processed like categories.

The next step was checking/identifying outliers and anomalies in the dataset **df_X_train_D**, with that in mind we calculated a z-score for each point and defined a threshold, those with a z-score greater than the threshold were considered outliers.

3- Data Processing

In data processing we prepared the data for modeling by cleaning it up and transforming it into a format that can be used by machine learning algorithms.

After the **Exploratory Data Analysis,** we are confident in our conclusions about the data analysis and applied them here in the data processing.

The first step was transforming the numeric columns ['D12', 'D13', 'D14', 'D15', 'D16', 'D17', 'D18', 'D19', 'D20', 'D21', 'D23'], into categorical columns, we achieved this using the "pd.get_dummies" function from the pandas library. Before applying this function we had to merge the "X_train" and "X_ivs" datasets in order to ensure the same number of columns in each.

The second step was after applying the "pd.get_dummies" function, the original numerical columns were eliminated from the merged dataset and afterwards the same dataset was divided into the "X_train" dataset and the "X_ivs" dataset again with the updated columns.

4- Feature Selection

We started our feature selection by verifying the correlations in our data. This step should be in the EDA, but we needed the data cleaned and prepared which means it needed to be after the **Data Processing** section.

We started by checking the correlation between each column and the target variable, and then how much each attribute correlates with each other, using the **df_X_train_D** and **df_X_train_FP** datasets for the D columns and the FP columns respectively. We concluded that the FP columns have a higher correlation with the target variable (y) and the D columns have a higher correlation between themselves.

The FP columns have sparsed values but they also have various points of correlation.

Before any feature selection the "X_train" (all columns), "X_ivs" (all columns), **df_X_train_D** and the **df_X_ivs_D** datasets were scaled, with the **MinMaxScaler,** FP columns were binary so they did not need scaling.

Then we did some testing for 3 different datasets, all the data, D columns only and the FP columns only datasets. The testing was divided into 4 parts:

- Feature selection with **SelectKBest**
- Random Forest Feature selection
- Stepwise Feature selection
- PCA for dimensionality reduction

We used PCA in an isolated manner and also followed by feature selection with **SelectKBest or Random Forest Feature selection.**

We experiment diverse kinds of approaches, to understand which are the best features and number of components to use when testing the regression models.

5- Evaluation

With the objective of providing the best possible regression model with cross validation.

We tested a combination of 1 scaler, 6 models, 6 feature selection and dimensionality reduction (PCA) methods, and 3 different numbers of features (31,50,75) based on the elbow point graph analysis (jupyter notebook) and an interval of values starting in 31 and finishing in 1500 for the number of components in PCA (having in consideration the percentage of total variance).

Regression Models:

- RandomForestRegressor()
- XGBRegressor()
- GradientBoostingRegressor()
- SVR()
- DecisionTreeRegressor()
- LinearRegression()

Scaler:

- MinMaxScaler()

Feature Selection and Dimensionality Reduction Methods:

- PCA followed by Random Forest Feature selection
- PCA followed by Feature selection
- Random Forest Feature selection
- Feature selection
- PCA
- Stepwise Feature selection

We tested the **XGBRegressor()** model as it helps preventing overfitting by incorporating the LASSO and Ridge regularization terms , it is fast and allows parallel processing that can be an advantage for a larger dataset like the one in this assignment, on the other and the **GradientBoostingRegressor()** model is less efficient in parallel processing making it slower as we confirmed in our testing.

We knew from the start that the **Stepwise Feature selection** was intrinsically slower than most feature selection methods in datasets with a relatively large amount of data, as it proceeds iteratively by changing base models and selecting the best alternative with the forward selection or backwards selection. After testing it we decided not to use it because of the long time it would take to analyze the best model in our dataset.

The evaluation metric used to evaluate the regression models performance was the **Mean Squared Error.** Below we have a table with the top 10 models.

Id	Name	N of components	N of features	Estimator	FS_Approach	Scaler	r2	MAE	MSE
0	All_PC400	400	75	SVR	pca_red_d	MinMax	0.6662	0.1213	0.0254
1	All_PC600	600	75	SVR	pca_red_d	MinMax	0.6637	0.122	0.0256
2	FP_PC400	400	50	SVR	pca_red_d	MinMax	0.6607	0.1224	0.0258
3	FP_PC600	600	50	SVR	pca_red_d	MinMax	0.6578	0.1232	0.026
4	FP_PC800	800	50	SVR	pca_red_d	MinMax	0.6549	0.1241	0.0263
5	FP_PC1000	1000	50	SVR	pca_red_d	MinMax	0.6516	0.1249	0.0265
6	FP_PC1250	1250	50	SVR	pca_red_d	MinMax	0.6482	0.1258	0.0268
7	All_PC800	800	75	SVR	pca_sel_f	MinMax	0.6481	0.1235	0.0268
8	All_PC400	400	75	SVR	pca_sel_f	MinMax	0.648	0.1234	0.0268
9	All_PC600	600	75	SVR	pca_sel_f	MinMax	0.6477	0.1233	0.0268
10	FP_PC1500	1500	50	SVR	pca_red_d	MinMax	0.6463	0.1263	0.0269

Table 1 – Top 10 classification models performance for number of components >= 400

Id	Name	N of components	N of features	Estimator	FS_Approach	Scaler	r2	MAE	MSE
0	FP_PC300_K31	300	31	SVR	pca	MinMax	0.6631	0.1216	0.0257
1	FP_PC300_K50	300	50	SVR	pca	MinMax	0.6624	0.1218	0.0257
2	FP_PC400_K31	400	31	SVR	pca	MinMax	0.66	0.1225	0.0259
3	FP_PC200_K31	200	31	SVR	pca	MinMax	0.6585	0.122	0.026
4	FP_PC200_K50	200	50	SVR	pca	MinMax	0.6567	0.1223	0.0261
5	FP_PC150_K50	150	50	SVR	pca	MinMax	0.6556	0.1222	0.0262
6	FP_PC150_K31	100	31	SVR	pca	MinMax	0.6548	0.1222	0.0263
7	FP_PC100_K31	100	31	SVR	pca	MinMax	0.6475	0.1234	0.0268
8	FP_PC100_K50	100	50	SVR	pca	MinMax	0.6467	0.1235	0.0269
9	FP_PC75_K50	75	50	SVR	pca	MinMax	0.6359	0.1254	0.0277
10	FP_PC75_K31	75	31	SVR	pca	MinMax	0.6351	0.1256	0.0278

Table 2 – Top 10 classification models performance for number of components <= 400

The best Regression Model obtained was the **Support Vector Regression (SVR()) model in the whole dataset (name = All), number of components = 400 with a total variance of 80,5%, number of features = 75 and with only dimensionality reduction (PCA), had a MSE =0.0254** as seen in table 1**.**

After analyzing the correlation in the feature selection section, both the total data (All) and the data with only the FP columns are sparse but have many points of correlation among them.

This type of correlation will leverage the good result of the linear combination of variables from PCA. These correlations will also contribute to the good results of the **SVR()** model with the linear kernel.

Using the number of components = 1500 in PCA will result in a high percentage of dimensionality reduction, approximately 90%. However, the best results are observed when the percentage of dimensionality reduction is between 70% and 80%, specifically with the number of components ranging from 300 to 400 as seen in table 1 and 2.

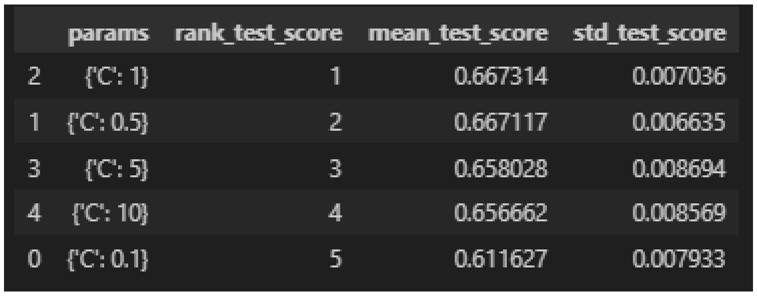
After analyzing the results, we noticed that the PCA alone or PCA plus a feature selection method sits at the top of the Feature selection approach, as well as the complete dataset (All) and the dataset with only the FP columns have better **MSE** values. The dataset composed by the D columns only sits right at the bottom of the scale with a **MSE** reaching almost **0.1.**

The second best regression model is as expected the **RandomForestRegressor()** model followed by **XGBRegressor()** model, these two models are not present in the top 10 list.

6- Model Tuning

After applying the **GridSearchCV** for the model tuning we found that the best hyperparameters are **C = 1** and **kernel = linear**, which makes sense as the C isn't either very large or very small providing the best balance between fitting the training data and avoid overfitting. If the value of the hyperparameter **C** was small the regularization term would dominate, and the model would be more tolerant of errors in the training data leading it to a smoother fit but with sacrifices on the accuracy, on the other hand if **C** was large the penalty for errors becomes more significant, and the model aims to fit the training data as accurately as possible increasing the risk of overfitting the model. About the **kernel** hyperparameter, the choice of kernel is crucial because it determines the shape of the decision boundary and influences how well the **SVR()** model can capture complex relationships in our dataset, and having in consideration the correlation between the features the linear kernel is the best option.

The **GridSearchCV()** can lead to overfitting of our model, having this in consideration we analyzed the GridSearch choices, as seen below.



	params	rank_test_score	mean_test_score	std_test_score
2	{'C': 1}	1	0.667314	0.007036
1	{'C': 0.5}	2	0.667117	0.006635
3	{'C': 5}	3	0.658028	0.008694
4	{'C': 10}	4	0.656662	0.008569
0	{'C': 0.1}	5	0.611627	0.007933

The choice with the best score is the C = 1, nevertheless the score is similar to C = 0.5 having in consideration that a smaller C is best for generalization we analyzed the MSE for both models. The model with the hyperparameter C = 1 gave us a MSE = 0.02532 and the model with the C = 0.5 gave us a **MSE = 0.02534**, as we can see the values are very similar again so we will choose, for the prediction the **SVR(C = 0.5) model** which will give us more generalization.