Introduction to Machine Learning Assignment 1

Bachelors

Spring

2024

1 General Instructions

Submission format. You are required to submit your solutions via Moodle as a single zip file. The zip archive should contain an Ipynb file, all data files as csv files, and model from the Task 1 named as **poly_optimized_model.sav**. Please, put your name and email at innopolis.university as the first line in the notebook. Source code should be clean, easy to read, and well documented - **poor code style will be penalized**, see the notebook head cell for the details. All the lines you should complete are marked with comments starting with *# TODO Write your code here.*

Points. The percentage near the Task Sections and Subsections denotes the maximum percentage of the assignment score that you can achieve. Bonus points can be awarded for elegant solutions, however, these bonus points will only be able to cancel the effect of penalties. You are going to solve 2 Tasks that will bring you the maximum score at best. We also provide a Bonus task (+10 points) as an airbag if you are not sure about your main solutions. Note that you cannot get more than 100 points.

Plagiarism policy. Plagiarised solutions will be heavily penalized for all parties involved. Also, do not just copy and paste solutions from the Internet. You are allowed to collaborate on general ideas with other students as well as consult books and Internet resources. However, be sure to credit the sources you use and type all the code and documentation by yourself.

2 Disclaimer

In this assignment, you are supposed to demonstrate the understanding and mastery of covered materials for the first 4 weeks. The scope of the assignment is data cleaning and preprocessing; training of classical machine learning algorithms; regression; classification; hyperparameter tuning; feature visualization, etc.
You might find task descriptions too clear and detailed, but it is done for the purpose: you should demonstrate that you can follow specific instructions to solve a machine learning task. Be ready that your next assignment will be more challenging and open-ended.
Since there are a lot of cell and instructions you need to do, it's easy to miss some of the subtasks, such as explanation of your observations. Please be careful when you read the descriptions, missed answers will be treated as zero.

3 Task 1 (50 %)

In Task 1 you should solve the multi-feature regression problem by training a LinearRegression model from sklearn. You are expected to be familiar with basic concepts of ML algorithm, such as training, performance evaluation, cross-validation, underfitting and overfitting, hyperparameter tuning, and feature visualization.

3.1 Linear regression (10 %)

A dataset is stored in file *train_1.csv* and contains 4 feature (X_{1-4}) and one label (y) columns. The nature of the columns is unknown for you, consider it as some non-linear function. This dataset does not require any preprocessing; however, you should split it into training and validation parts.

1. Load the data set from train_1.csv file as a Pandas DataFrame.
2. Split and shuffle into training (80% of the entire dataset) and validation (20% of the entire dataset) DataFrames.
3. Using the **linear regression model** from sklearn, fit the model to the training dataset.
4. Evaluate on the validation dataset by printing MSE, RMSE, MAE and R2 score.

3.2 Tuning of polynomial regression - (25%)

Let's try to fit a polynomial regression model to the same dataset and compare the performance. In polynomial regression, you are going to deal with 1 hyperparameter per feature, a polynomial degree. Because we don't know what is the correct degree, we have to tune this value using Grid Search algorithm.
In this subtask, your maximum score depends on the evaluation of the **closed** test data set, which is not given to you. We guarantee that the test data are randomly sampled from the same distribution as the training one. Points distribution:

- RMSE $>$ 65 : 2 points;
- 65 $\geq$ RMSE $>$ 20 : 5 points;
- 20 $\geq$ RMSE $>$ 5 : 7 points;

- $5 \geq$ RMSE > 1.4 : 9 points;
- RMSE ≤ 1.4 : 10 points.

You are supposed to do the following:

1. Using the sklearn pipeline, construct a polynomial regression pipeline consisting of a (1) polynomial feature class and (2) a linear regression class. Use "degree=2" in polynom.
2. Use GridSearch to find the best polynomial regression model and print the best parameters. Set *degrees = range(2, 6), cross-validation=8, scoring ='negative mean squared error'*, using the best params, predict on "X_test" and evaluate using MSE, RMSE, MAE and R2 score.
3. **Save the model** without changing a name of file (poly_optimized_model.sav). This model will be used for your score calculation in this task.

3.3 Determine the linear dependent features (15%)

Regression algorithms are based on the assumption of independence of features. Since we have several features, it is interesting to determine if *some of the features are correlated*. The data set contains at least one such linear dependent pair, and in this task you are asked to determine it and explain your choice. We recommend to use feature visualization (PairGrid plot).

4 Task 2 (50 %)

In this task, you will be solving a binary classification task. You will be classifying a Pokemon whether it is a legendary Pokemon or not. Besides the classification, you should be familiar with concepts of data preprocessing, classification metrics, regularization, etc.
Information about the dataset (pokemon_modified.csv):

- There are 36 columns that represent the features of pokemon.
- There are 801 rows in the dataset, each row is encoding a pokemon.
- The label column is "is_legendary" which tells you if the pokemon is legendary or not.
- There are 3 feature columns that are missing some values.
- There are 2 useless for classification feature columns.
- There is 1 feature column that should be categorically encoded.

4.1 Data preprocessing (20 %)

In this section you have to load the data, analyze the features, decide which features should be removed, what should be encoded, and what should be imputed.

1. Load the pokemon dataset using pandas dataframe.
2. Explore the dataset using pd.head(), pd.info(). Remove the redundant 2 features and explain why they should be removed.
3. Split the dataset into train/test with a ratio of 0.8/0.2. Is the dataset balanced in terms of classes?
4. Check for missing values and impute them using the SimpleImputer. You can use *mean* or *most frequent* strategies.
5. Double check that there are no missing values.
6. Identify and encode the categorical feature using the OneHot encoder.
7. Scale the data using MinMax normalization or StandardScaler.
8. Plot the correlation matrix. Answer the following questions: Are there highly correlated features in the dataset? Is it a problem? Preprocess data if necessary.

4.2 Model fitting and comparison - (30 %)

You have to train and evaluate several classification models from sklearn:

- Logistic Regression classifier;
- KNN classifier;
- Gaussian Naive-Bayes classifier.

There are hyperparameters for logistic regression and KNN which we need to tune using Grid-Search: inverse of regularization strength C , the regularization technique, and the solver.
The hyperparameter of the KNN is the K value which is the number of neighbors, the metric which is the distance computation function and the weight, whether to give all the neighboring points the same weight or give a weight based on the distance.
Naive-Bayes classifier doesn't require tunning of hyperparameters.
Follow the instructions:

1. Using GridSearchCV, find best hyper-parameters for the Logistic Regression model. Try different variations with *penalty: ['l1','l2']; C: np.logspace(-3,3,7); solver: ['newton-cg', 'lbfgs', 'liblinear'], scoring: 'f1'*.
2. Evaluate the logistic regression model with the best parameters on the test data using accuracy, precision, recall, and F1 score.
3. Print the regression coefficients and find the names of the top 5 most influencing features and the top 5 ignored features.
4. Using GridSearchCV, find best hyper-parameter for the KNN model. Try different variations with *k :list(range(1, 15)), weights:['uniform', 'distance'], metric:['euclidean', 'manhattan', 'chebyshev', 'cosine']*.
5. Evaluate the KNN model with the best parameters on the test data using accuracy, precision, recall, and F1 score.
6. Fit Gaussian Naive-Bayes to the data and evaluate on the test dataset while printing the metrics.
7. Which metric is most appropriate for this task and why? Explain what kind of error (false negative or false positive) is more critical for this data set.
8. Compare the 3 classifiers in terms of accuracy, precision, recall and F1-score. What is the best model for this task? Explain why.

5 Bonus task (10 %)

The purpose of the Bonus task is twofold: (a) to facilitate further learning, and (b) to compensate for any potential errors made in previous tasks. We expect that you will search the information about this topic by yourself. Previously, you implemented a binary classifier to classify if a pokemon is legendary or not. In this task, you will take this concept one step further solve a multi-class classification problem. There are two techniques where you can solve this problem: One-vs.-Rest and Multinomial classifier.

The dataset contains 3 features and 1 outcome variable with 3 classes. The dataset is clean and preprocessed for you and there are no missing values.

1. Load the data from bonus_train.csv and bonus_test.csv using pandas dataframe and then split the training data to X_train and y_train, split the test data to X_test and y_test.

2. Using seaborn library, plot the training data using the pairplot with *kind = "scatter"* and *hue="target"*.

3. Using Logistic regression model from sklearn, fit the model to the training data, first by using the multinomial technique and second by using one-vs.-rest.

4. Evaluate the Logistic Regression models using the mean accuracy on the test dataset.

5. Use GridSearch to tune the C value of the logistic regression, and the multi class. Use the *C: np.logspace(-10, 10,7), multi_class : ['multinomial', 'ovr']*

6. Comment on why one was better than the other.

7. Lastly visualize the decision boundary of the best performing model on the training dataset in 2D. Hint: fit the model on just two features and plot.