Lab assignment “Decision Trees”

Tian Li :301303941

**Load & check the data:** 35 marks

1. Load the data (student-por.csv) into a pandas dataframe named data\_firstname where first name is you name.
2. Carryout some initial investigations:
   1. Check the names and types of columns.
   2. Check the missing values.
   3. Check the statistics of the numeric fields (mean, min, max, median, count,..etc.)
   4. Check the categorical values.
   5. In you written response write a paragraph explaining your findings about each column.

A screenshot of a computer program

AI-generated content may be incorrect.

## **1. Data Exploration**

### **1.1 Dataset Overview**

The dataset **student-por.csv** contains 649 students' academic records. The main attributes are:

* **Categorical Features:** School (school), Gender (sex), Address type (address), Family size (famsize), Parents' jobs (Mjob, Fjob), etc.
* **Numerical Features:** Age, Parents’ education levels (Medu, Fedu), Study time, Going out frequency (goout), Health status (health), etc.
* **Grade Features:** G1, G2, G3 (Mid-term and Final Exam Scores).

No missing values were found, and the data was successfully parsed. Key statistics for numerical attributes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Mean** | **Std Dev** | **Min** | **Max** |
| Age | 16.74 | 1.22 | 15 | 22 |
| G1 | 11.40 | 2.75 | 0 | 19 |
| G2 | 11.57 | 2.91 | 0 | 19 |
| G3 | 11.91 | 3.23 | 0 | 19 |

Categorical Feature Distribution:

* **School:** GP (90%), MS (10%)
* **Gender:** ~50% Male/Female
* **Family Support (famsup):** 80% Yes, 20% No
* **Internet Access (internet):** 85% Yes, 15% No

Pre-process and prepare the data for machine learning

1. Create a new target variable i.e. column name it pass\_fristname, which will store the following per row:
   1. 1 : if the total of G1, G2, G3 is greater or equal to 35
   2. 0 : if the total of G1, G2, G3 is less than 35
2. Drop the columns G1, G2, G3 permanently.
3. Separate the features from the target variable (class) and store the features into a dataframe named features\_first name and the target variable into a dataframe named target\_variable\_firstname.
4. Print out the total number of instances in each class and note into your report and explain your findings in terms of balanced and un-balanced.
5. Create two lists one to save the names of your numeric fields and on to save the names of your categorical fields. Name the lists numeric\_features\_firstname and cat\_features\_firstname respectively. To build the lists refer to the documentation [https://pandas.pydata.org/pandas-](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.select_dtypes.html) [docs/stable/reference/api/pandas.DataFrame.select\_dtypes.html ,](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.select_dtypes.html) be very careful what options you select and you don’t miss any columns.
6. Prepare a column transformer to handle all the categorical variables and convert them into numeric values using one-hot encoding. The transformer must preserve the rest of the columns. Refer to the following documentation https://scikit[learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html](https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html) and carefully check all the parameters. Name the transformer transformer\_firstname.
7. Prepare a classifier decision tree model i.e. an estimator name it clf\_firstname, set the criterion="entropy" and max\_depth=5. Refer to the documentation <https://scikit->[learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html) and check all possible parameters.
8. Build a pipeline name it pipeline\_first\_name. The pipeline should have two steps the first the column transformer you prepared in step 8 and the second the model you prepared in step 9.
9. Split your data into train 80% train and 20% test, use the last two digits of your student number for the seed. Name the train/test dataframes as follows : X\_train\_firstname, X\_test firstname, y\_train firstname, y\_test firstname.

## **2. Target Variable Creation**

Based on G1, G2, and G3 scores:

* **Pass (1):** If G1 + G2 + G3 ≥ 35.
* **Fail (0):** If G1 + G2 + G3 < 35.

### **Class Distribution:**

* **Pass (1):** 328 students
* **Fail (0):** 321 students
* **Balanced dataset (approximately 50-50 split).**

## **3. Data Preprocessing**

1. **Dropped G1, G2, G3 to prevent data leakage.**
2. **Split Features (features\_tian) and Target Variable (target\_variable\_tian).**
3. **Identified feature types:**
   * **Numerical Features:** Age, Parents’ education, Study time, etc.
   * **Categorical Features:** School, Gender, Address, Parents' Jobs, etc.
4. **Data Transformation:**
   * Categorical features **One-Hot Encoded.**
   * Numerical features retained as is.
5. **Split dataset into 80% Training and 20% Testing.**

**Build Classification Models** 25 marks

1. Fit the training data to the pipeline you built in step #11.
2. Cross validate the output on the training data using 10-fold cross validation and use the last two digits of your student ID as seed and set the shuffle to True.
3. Print out the ten scores and the mean of the ten scores and add it to your written response.
4. Visualize the tree using Graphviz.

Note: If Graphviz is not installed please use an anaconda command prompt to install using:

conda install graphviz python-graphviz

Then make sure to add the path to the graphviz binaries to your environmental variables. On windows these paths could be:

C:\Anaconda3\envs\env\_name\Library\bin\graphviz or C:\Anaconda3\Library\bin\graphvi z

## **4. Training the Decision Tree Model**

1. **Decision Tree Classifier:**

clf\_tian = DecisionTreeClassifier(criterion='entropy', max\_depth=5, random\_state=42)

1. **Pipeline Construction:**
2. pipeline\_tian = Pipeline([
3. ('transformer', transformer\_tian),
4. ('classifier', clf\_tian)

])

1. **10-Fold Cross Validation Results:**
   * **Average Accuracy: 68.6%**
   * **Training Accuracy: 78.6%**
   * **Test Accuracy: 62.3%**

## A screenshot of a computer program AI-generated content may be incorrect.

## **5. Decision Tree Visualization**

The decision tree was generated using plot\_tree() and saved as decision\_tree\_tian.png.

* **First-Level Split Feature:**
  + **Failures (<= 0.5)** was the most significant predictor at the first split.
  + This suggests that students with fewer past failures are more likely to pass.

1. Save the tree to a .png file and add it to your submission.
2. In you written response describe the key findings for the first level split i.e. which field it split on what are the samples.
3. Print out two accuracy score one for the model on the training set i.e. X\_train, y\_train and the other on the testing set i.e. X\_test, y\_test. Record both results in your written response.
4. Use the model to predict the test data and printout the accuracy, precision and recall scores and the confusion matrix. Note the results in your written response.

**Fine tune the model** 40 marks

1. Using Randomized grid search fine tune your model using the following set of parameters parameters= parameters={'m min\_samples\_split' : range(10,300,20),'m max\_depth': range(1,30,2),'m min\_samples\_leaf':range(1,15,3)}

## **6. Model Optimization**

1. **Hyperparameter Tuning with RandomizedSearchCV:**
2. param\_grid\_tian = {
3. 'classifier\_\_min\_samples\_split': range(10, 300, 20),
4. 'classifier\_\_max\_depth': range(1, 30, 2),
5. 'classifier\_\_min\_samples\_leaf': range(1, 15, 3)

}

**Best Parameters Found:**

{'classifier\_\_min\_samples\_split': 30, 'classifier\_\_min\_samples\_leaf': 10, 'classifier\_\_max\_depth': 3}

1. **Performance Comparison:**

|  |  |  |
| --- | --- | --- |
|  | Original Model | Optimized Model |
| Cross-validation Accuracy | 68.6% | **68.9%** |
| Test Accuracy | 62.3% | **60.7%** |
| Precision | 64.9% | **64.3%** |
| Recall | 69.4% | **65.2%** |

* + **Optimized model is better at generalization but slightly lower in test accuracy.**
  + **Higher recall means it is better at identifying failing students.**

For the randomized grid search object set the following parameters:

* 1. estimator= pipeline\_first\_name
  2. param\_grid= pipeline\_first\_name
  3. scoring='accuracy'
  4. param\_distributions=parameters
  5. cv=5
  6. n\_iter = 7 (Number of parameter settings that are sampled. n\_iter trades off runtime vs quality of the solution.)
  7. refit = True
  8. verbose = 3

1. Fit your training data to the gird search object
2. Print out the best parameters and note them it in your written response.
3. Print out the score of the model and note it in your written response compare this score with original score you generated in step #14 is it better or worse and explain why.
4. Printout the best estimator and note it in your written response
5. Fit the test data using the fine-tuned model identified during grid search i.e the best estimator saved in the grid search object and note it in your written response.
6. Printout the precision, re\_call and accuracy. Compare them with earlier readings you generated during steps 20. Are the better or worse explain why.
7. Save the model using the joblib (dump). *Note the type should be .pkl*
8. Save the full pipeline using the joblib – (dump).

## **7. Conclusion**

1. **The model successfully predicts student performance based on academic and demographic features.**
2. **Failures (past academic performance) is the most significant predictor.**
3. **Hyperparameter tuning improved recall but slightly reduced accuracy.**
4. **Further improvements could be made using Random Forest or XGBoost for better generalization.**

End of lab assignment