FINAL PROJECT

Part A & Part B Report

KAIJIE “YURI” YU

PATRICK WATKINS

IRMA COVARRUBIAS

Center for Information Systems & Technology

Claremont Graduate University

IST 340: Knowledge Discovery & Data Mining

May 12, 2023

Table of Contents

Should there be any discrepancies, the Jupyter notebook should be taken as the authoritative source.

Part A

Task A1: Best Model Recommendation

A1.1 DESCRIPTION OF PLAN

- Business Objectives Determination

Background

The Data Mining Project Team has been tasked with developing and building a predictive model to assess the risk associated with student loan applicants. Specifically, the Project Team is to present an accurate predictive model that will help identify those student characteristics considered a "good loan risk" (positive) and a "poor loan risk" (negative). The loan operations team is requesting this model in order to gain valuable insights on student customers and future applicants, make better data-driven and informed decisions, manage risk, minimize financial losses, and improve overall loan operations.

The Data Mining Project Team is comprised of:

- Project Leader: Ms. Imani Kuhn
- Several end-users from loan operations
- Data Mining Analytical Experts: Kaijie "Yuri" Yu, Patrick Watkins, Irma Covarrubias

With her graduate background in Management Science, Ms. Imani Kuhn is the project lead and will have final approval of any recommended models made to the Project Team before model deployment.

Business Objectives

Using a very accurate predictive model to make a significant impact on the operations, which may classify good loan risks based on features and behavior captured during the application process.

- Data Understanding

Collect Initial Data

Read the content in the prolog and Excel files and save them into data frames.
The data sources were provided on shared drive at the following link: https://cgu.instructure.com/courses/11475/files/folder/Student%20Documents/Final%20Project%20Documents/Datasets. From this link, we were able to acquire, access, and download the following datasets (Table 3):

Data Source / File Name
no_payment_due.pl
male.pl
Region, Marital Status.xlsx

Longest_Absense_From_School.pl
Family_Income.xlsx
enrolled.pl
enlist.pl
unemployed.pl
filed_for_bankrupcy.pl
disabled.pl
Parents_Education.xlsx
LoanAmount.xlsx
CreditRating Data 2012.xlsx
HS_Academics 2012.xlsx
Personality Characteristics Data.xlsx

Table 3: List of data sources for project

Once the data sources are downloaded, they will be uploaded to Project Team's GitHub repository and Google Drive. From the repository, these files can be saved to Google Drive where they can be read/loaded into Google Colaboratory (Colab) and converted into Pandas data frames. https://github.com/MSWinds/dm-group1-project/tree/main/datasets

Describe Data

The data sources were provided as either an Excel spreadsheet (.xlsx) or Perl script (.pl). Each data source was loaded and converted into Pandas data frames in Google Colab. The data does satisfy and meet the relevant data requirements specified in the project description. The next two tables show our initial data collection report and the field description and requirements from the project description (Table 4 and 5).

Data Source	Field Name(s)	Total Fields	Total Records
no_payment_due.pl	StudentId, NoPaymentDue (pos / neg)	2	1000
male.pl	StudentId	1	306
Region, Marital Status.xlsx	StudentId, Country, Region, Marital_Status	4	95
Longest_Absense_From_School.pl	StudentId, Number_of_Months	2	98
Family_Income.xlsx	StudentId, Family_Income	2	1000
enrolled.pl	StudentId, School, Units	3	1000
enlist.pl	StudentId, Service	2	497
unemployed.pl	StudentId	1	1000
filed_for_bankrupcy.pl	StudentId	1	96
disabled.pl	StudentId	1	1000
Parents_Education.xlsx	StudentId, Parents' Education Level	2	1000
LoanAmount.xlsx	StudentId, Loan	2	1000
CreditRating Data 2012.xlsx	StudentId, Credit_Score_Raw, Credit_Score_AgeAdj	3	1000
HS_Academics 2012.xlsx	StudentID, HS_Math, HS_Science, HS_English	4	1000
Personality Characteristics Data.xlsx	StudentID, LongTermPlannerScore, DecisionStyle, PowerOrientedScore, CommunityOrientedScore	5	1000

Table 4: Initial Data Collection Report

Field Name	Possible Model Role	Required Data Type
StudentID	ID	Interval
NoPaymentDue	Target	Binary
Enlisted	Input	Binary
Service	Input	Nominal
Disabled	Input	Binary
Unemployed	Input	Binary
Enrolled	Input	Binary
School	Input	Nominal
Units	Input	Interval
Region	Input	Nominal
Country	Input	Nominal
Longest_Absence_From_School	Input	Interval
Gender	Input	Binary
Marital_Status	Input	Binary
Filed_for_Bankrupcy	Input	Binary
LongTermPlanningScore	Input	Interval
DecisionStyle	Input	Nominal
PowerOrientedScore	Input	Interval
CommunityOrientedScore	Input	Interval
Family_Income	Input	Interval
Parents_Education_Level	Input	Ordinal
Credit Score_Raw	Input	Interval
Credit Score_AgeAdj	Input	Interval
Loan Amount	Input	Interval
HS_Math	Input	Ordinal
HS_Science	Input	Ordinal
HS_English	Input	Ordinal

Table 5: Field Description & Requirements from Project Description

In comparing Table 4 and 5, here is an initial list of some problems encountered based on an initial analysis of each data source:

- All the data sources will need to be joined on the StudentID unique identifier.
    - For this to happen, StudentID will need to be standardized across all data sources:
        - StudentID must be spelled the same way across all data sources.
        - StudentID must be the same data type across all data sources (int)
- All field names must match the field names specified in the project description (Table 5).
- The total unique records are 1000, but some of the data sources have less than 1000, so thus, some categories are missing. For example, the male.pl a list of student males and their StudentIDs. In this case, we will have to join on the StudentID, identify those records as "Male" and identify all others as "Female."
- When dealing with different data sources and different file types, we will have to ensure that data types reflect correctly. Because Python does not always recognize the data type, it will default to data type "Object," so we need to check every column.
- We will have to review each data frame to see if there are any duplicate records, 1:M relationships, and missing data.
- We will have to encode some of the fields to match the relationships outlined in the project description shown here (Table 6):

Field Name	Data Source	Value
StudentId	No_payment_due: StudentId	
NoPaymentDue	No_payment_due: NoPaymentDue	
Enlisted	Enlist	**1** if the corresponding StudentId is in the **Enlist** file; **0** otherwise
Service	Enlist: Organization	
Enrolled		**1** if the corresponding StudentId is in the **Enrolled** file; **0** otherwise
School	Enrolled: School	
Units	Enrolled: Units	Sum *
Disabled	Disabled:	**1** if the corresponding StudentId is in the **Disabled** file; **0** otherwise
Unemployed		**1** if the corresponding StudentId is in the **Unemployed** file; **0** otherwise
Filed_for_Bankrupcy		**1** if the corresponding StudentId is in the **Filed_for_Bankrupcy** file; **0** otherwise *
Longest_Absence	Longest_absense_from_school: Number_of_Months	
Male		**1** if the corresponding StudentId is in the **Male** file; **0** otherwise
Marital Status	Region, Marital Status xlsx	**1:** Married **0:** Single
Region	Region, Marital Status xlsx	**1, 2, 3, 4, 5**
Parents' Education Level	Parents_Education.xlsx	'VERY LOW', 'LOW', 'MEDIUM', 'HIGH', 'VERY HIGH'

Table 6: Field and Data Source Relationships as specified in the Project Description

Based on the initial analysis, once these issues listed are addressed, we can merge all data frames into one (1) large single data frame. It is also important to point out that only two (2) of the data sources mention 2012. Thus, we are assuming that the rest of the data sources are also from 2012.

Explore Data (Data exploration)

For this section, we plan on exploring the metadata, descriptive statistics, distribution, correlation, and visualization for each data source. Also, we plan on examining the relationships between financial variables in relation to the target variable. We will start thinking about the following questions:

- What derived variables can we possibly obtain from the data sources?
- For categorical variables that have more than 2 unique values, are there other numerical data we can substitute for?

Verify Data (Data exploration)

To verify data quality, we will examine every variable for missing values, duplicates, misspellings, counts vs distinct values, check for outliers, and check for errors. Also, we will establish clear relationships between the variables and the target events to understand the results. For any data quality issues discovered, we will note them, engage the end-users on these anomalies, and list possible solutions for each issue.

- ## Data Preparation

Clean Data

- Use proper missing value imputation method.
- Change the “StudentID” field to “int” type from “object” type. This applies to the Prolog files.
- Check the duplicates and remove the duplicates.
- Detect the outliers.

Integrate Data (merged_raw_data.csv)

- merge all the data sets.
- for enlisted and enrollment data sets, aggregated units from different schools, and concatenated service and school names as a single string separated by a semicolon.

Format Data

- Other than “StudentID”, change all fields containing numerical data to the “float” data type.
- Change the NaN values to None or 0 based on the rules in Table 3.
- Group the categories if necessary.

Construct Data

- one-hot encoding, ordinal encoding, or label encoding.
- Remove the non-predictor variables for the classification model.

- ## Data Mining

Modeling

- Modeling Techniques
- Test Design – 70% Training 30% Testing
- Build Model – Logistic Regression, Decision Tree, Random Forest, gb, mlp – 8 models
- Hyperparameter tunning –grid search, random search, manually tunning
- Assess Model – Accuracy, lift, stability, overall score(probably)

- ## Evaluation

- Evaluate results – Choose the best model that has the highest accuracy while maintaining other good scores.
- Review Process – describe the best model

- Create and Apply Score Dataset

Select the 35 rows of variables that are required for the predictive model, except the target variable.

A1.2 EXECUTION OF PLAN

- Data Understanding

Collect Initial Data

First, all 15 datasets were read and put into their own individual data frame. The data type of column "StudentID" was changed to "int" in every data frame. Other columns containing numerical data besides StudentID were changed to the float data type.

Describe Data

A metadata data frame was created to describe information of the input data. This information included data type, number of missing values, missing values percentage, number of unique values, unique values percentage, and basic statistics listing the minimum, maximum, and standard deviation.

Explore Data

Exploratory analysis was performed on a specified column in each of the 15 data frames that displayed basic statistics and visualizations. The visualizations included bar plots, box and whisker plots, and/or histograms. Depending on the plot, counts and percentages of the unique values or mean, standard deviation, and number of outliers were displayed.

Merge the raw files

For the service and school data frames, we aggregated the information into a single record for each student. We created new columns to capture the total number of services or schools attended and concatenate the service and school names as a single string separated by a delimiter. This way, we will maintain the specific information without duplicating student records.

Describe and explore merged data

Other than the HS_English, we will need to format the categorical data. Missing values will be marked as other category. The missing value for units will be marked as 0 since those students have not attended any school yet nor taken any units. For school and service, we will need to use one-hot encoding to convert the categorical data into numerical data with new columns.

Conclusion of the data understanding phase

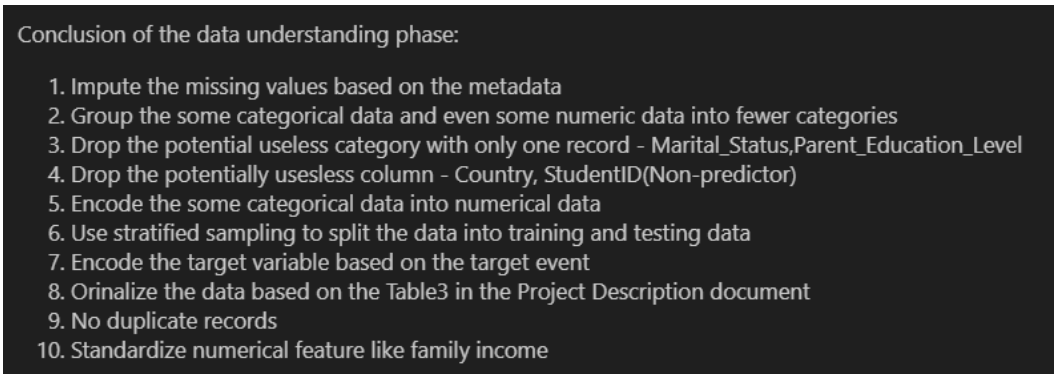


Figure 1: Data Understanding Summary

- ## Data Preparation

Select Data

- Drop the 'Country' column. There is one record that is from Belize, if we drop this record all of the other records are from the United States, rendering a 'Country' column unnecessary.
- Drop the 'StudentID' column because it is not necessary for modeling.
- Since category 2 only has one record in Marital_Status, drop this deviant record.
- Drop the record that has Parent_Education_Level= 'GREATE'.

Clean Data

- In Parent_Education_Level, group 'HS or lower' and 'VERY HS or Lower' together. This column will eventually be encoded to ordinal.
- The column 'Loan Amount' has a lot of outliers that roughly form a group, we will group data into two categorical groups. If the Loan_Amount is less than 12000, we will group it into 0. Otherwise, we will group it into 1.
- Replace the missing values with 0 in the following columns: 'Units', 'Disabled', 'Unemployed', 'Gender', and 'Filed_for_Bankruptcy'.
- There are only 1.6% missing values in the 'HS_English' column, and it is right-skewed. Replace the missing values with the median. The median is 8.0.

Construct Data

- Perform one-hot encoding on the 'School' and 'Service' columns because of the aggregated values.
- Add a 'School' and 'Service' column to the appropriate one-hot encoded columns.
- Concatenate the original data frame with the one-hot encoded columns.
- Drop the original 'School' and 'Service' columns.
- Save a copy of the cleaned data set.

Format Data

- Ordinalize the data:

    - Ordinalize Encoding for Parent_Education_Level: HS or Lower -> Bachelor's -> Master's -> Doctorate.
    - Convert the 'Longest_Absence_From_School' column from object to numeric.
- Binary Encoding:
    - Use binary encoding for the 'DecisionStyle' column in case there are some models that require numeric data.
- Normalize the Data:
    - Create a scalar object, drop the original 'Family_Income' column, and then normalize the column.

Data understanding and data preparation issues

- We discussed the 'service' and 'school' columns issues since they have one-to-many relationships. From domain expert's feedback, we chose to use one-hot encoding.
- We discussed the dataset that might not be suitable for all kinds of models.
- The tree-based algorithms (DT, XGB, RF) can handle all types of measurements (Binary, Nominal, Interval, Ordinal) well without the need for one-hot encoding.
- Other models can handle binary, interval and ordinal data well after proper preprocessing but require one-hot encoding or other appropriate methods for nominal data.
- Therefore, we will use the tree-based algorithms for the initial model with df_a1. And then, we need to do some extra data preprocessing for the other models.
- Tree-based algorithms: DT, XGB, RF
- Other models: LR, SVM, KNN, NB, MLP

• Experimental Approach

Encode Target Variable for A1

Since the target event is NoPaymentDue = 'pos', we will encode the target variable to 1 if NoPaymentDue = 'pos' and 0 otherwise for df_a1

Data Splitting

The data set was split into 70% training and 30% testing with stratified sampling used.

Modeling

- Decision Tree, XG Boost, Random Forest, KNN, Logistic Regression, MLP, MNB, and SVM models were built.
- Hyperparameter tuning – Grid search was used for the DT, KNN, and MLP models, while a random search was used for RF.

Extra Data Preparation for the other models

- We used one-hot encoding for Region since it is a numeric nominal data.

- Before we use the logistic regression, we need to check the several assumptions of the logistic regression. Two assumptions needed to be check, which are:
- There is no multicollinearity among the independent variables.
- Linearity of independent variables and log odds.
- And our conclusion was that logistic regression can still work reasonably well even if some of the assumptions are violated.

• Definition of performance measures

The most important performance measure in A1 was accuracy, however, classification reports were still conducted to see the precision and recall as well. Stability and lift (at the third decile) were also calculated.

Measure	Description / Definition of Value Function
Accuracy	1- Misclassification Rate
Precision	TP/(TP+FP)
Recall	Recall = TP/(TP+FN)
Stability	Based on the slope at the third decile through visual inspection. A decrease in rate or no change in the rate indicates stability.
Lift	(TREECAPC-BASECAPC)/(BESTCAPC-BASECAPC)
F1	(2*Precision*Recall)/(Precision+Recall)
AUC_ROC	the area under the curve

• Definition of approach for evaluating generated models

As instructed by the prompt in Task A1, the model with the highest accuracy was chosen as the best model. We also checked the other metrics mentioned above in the definition of performances to make sure the chosen model still performed well in other areas.

	Model	Accuracy	Precision	Recall	F1	AUC_ROC	Lift_Score	Stability
0	SVM	0.986700	0.994800	0.984500	0.989600	0.999500	1.000000	1
1	DT	0.973300	0.969500	0.989600	0.979500	0.996400	0.906566	1
2	KNN	0.796700	0.907400	0.761700	0.828200	0.880600	0.719637	0
3	XGB	0.980000	0.974600	0.994800	0.984600	0.999700	0.875421	0
4	MNB	0.953300	0.973500	0.953400	0.963400	0.988300	0.875421	0
5	MLP	0.956700	0.941200	0.994800	0.967300	0.998100	0.875421	0
6	RF	0.973300	0.969500	0.989600	0.979500	0.998300	0.906566	0
7	LR	0.996700	1.000000	0.994800	0.997400	0.999900	1.000000	1

Logistic Regression has the best performance among all the models as it has a very high accuracy and also mantains high values for other metrics. Thus the logistic regression model is selected as the best model for Task A1.

The features in your dataset are already on a similar scale or their ranges are not drastically different, then normalization might not be as crucial for Logistic Regression. In such cases, the algorithm can still converge and find a good solution.

Besides, the dataset meets most of the assumptions of the logistic regression while violating some of them slightly. And those variables that violate the assumptions are not significant in the model.

For this task A1, the goal of a machine learning model isn't to strictly satisfy every mathematical assumption or condition, but to make accurate and useful predictions on new, unseen data while meeting the requirments from the customer/end-user/project manager. Besides, the tree-based models are having good performances as well.

Therefore, we will choose the **logistic regression** as the best model for Task A1.

Figure 2: Results Table and Summary

- Score Dataset

A score dataset that consists of 35 new cases that were not in the original dataset was then generated, and the chosen model was applied to the new dataset.

Task A2

A2.1 DESCRIPTION OF PLAN

- Business Objectives Determination

Generate an explanatory predictive model, preferably a Decision Tree with 4 to 6 rules, that accurately finds the good loan risk students.

- Evaluation

Since the project team thought that it would be better to have an explanatory model. We can pick the decision tree model as the best model for Task A2. Let's first check the structure of the tree to see if it is easy to interpret and meet the requirement.

- Describe and interpret the best model

Try to describe the decision tree model we picked.

- Create and Apply Score Dataset

Score dataset is different from the one in Task A1. This score dataset will use the nominal column "Region" for our decision tree model.

A2.2 EXECUTION OF PLAN

- Definition of performance measures

Same as Task A1, but now we add Simplicity into the performance measures.

Measure	Description / Definition of Value Function
Simplicity	Number of leaves is between ideal top and bottom cutoffs. In this case, the tree should be within 4 to 6 rules.

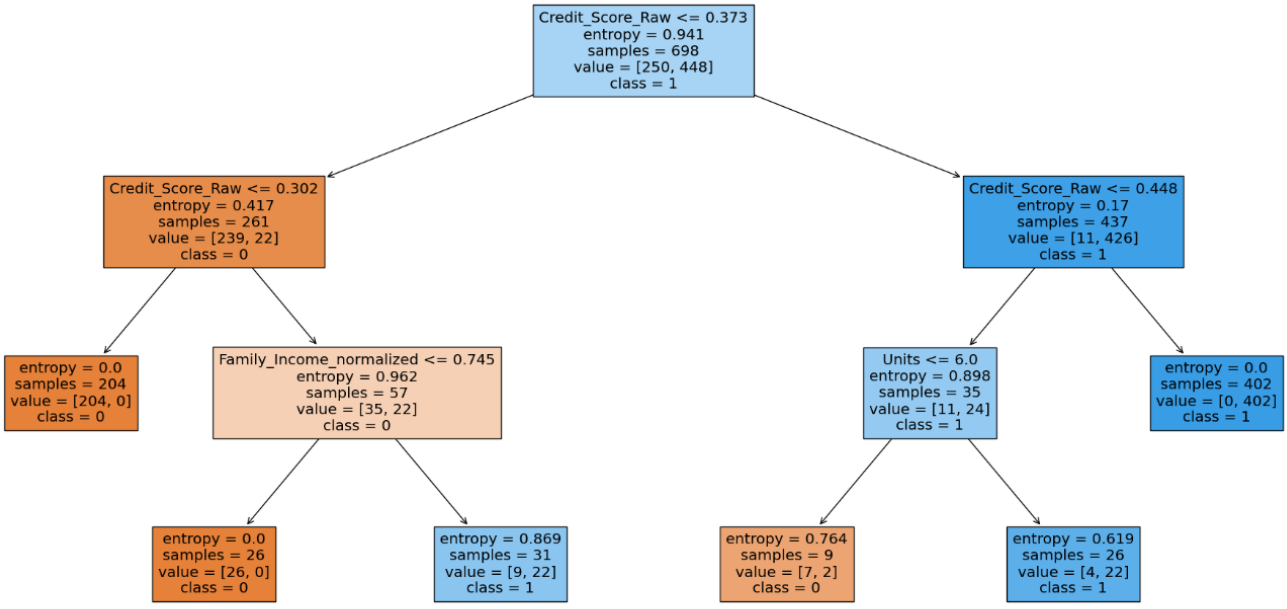
- Definition of approach for evaluating generated models

- The generated model was a decision tree model.
- The previous metrics of accuracy, stability, precision, recall, and lift were evaluated. In addition to the metrics used in Task A1, simplicity of the decision tree model was also evaluated. The model had a leaf size of 6, which was in the ideal leaf range listed. This leaf size yielded a simplicity value of 1.
- The ideal leaf size is set between 4 and 6 cut off leaf sizes are <=3 or >=9.
- The leaf size of the DT model is 6, which is within the ideal range. So the simplicity of the model is 1, which is good.

	Model	Accuracy	Precision	Recall	F1	AUC_ROC	Lift_Score	Stability	Simplicity
1	DT	0.9733	0.9695	0.9896	0.9795	0.9964	0.906566	1	1

Figure 3: DT results

Therefore, this decision tree is selected as the best model for Task A2.



- <u>Interpret the model</u>

The decision tree is using "Credit_Score_Raw" as the most important feature. The tree splits the data into two groups based on whether the credit score is above or below 0.373. The tree then uses other features like "Family_Income_Normalized" and "Units" to further split the data into smaller groups. The final decision for each group is based on the majority class in the corresponding leaf node.

For example, If Credit_Score_Raw<=0.373, Credit_Score_Raw>0.302, Family_Income_Normalized>0.745, the majority class will be 1, which means the student has no payment due(NoPaymentDue='pos'). It is logically making sense since the student has a high credit score and high family income, which means the student is more likely to have no payment due.

For Family_Income_Normalized, larger than 0.745 means the student has a high family income since we used MinMaxScaler to normalize the data. The threshold can be calculated by using the formula below: threshold_income = (normalized_threshold * (max_income - min_income)) + min_income = 85,352, which is higher than the mean of the family income.

Figure 4: Model interpretation

- Score Dataset

Like Task A1, a score dataset was used to assess the model, however, the dataset was not the same. This score dataset used the nominal column 'Region' in the score dataset.

TASK A3

A3.1 DESCRIPTION OF PLAN

- Business Objectives Determination

- Recommend a clustering model that can segment applicants and borrowers into different groups or customer personas based on student creditworthiness and other factors to tailor loan product offerings and marketing efforts to meet the needs of various customer segments.
- Obtain the highest lift values with the model of at least 30% or greater.
    - Top 30% of loan cases

- Evaluation

- Whichever model has the highest lift value at the third decile will be the model of choice for this task. Examine the other performance measures and see if they are still of quality.
- Since we have already built the model in Task A1, which meets the requirements of Task A3, we will assess the model and apply the model to the score dataset.
- To satisfy that requirement, the most appropriate predictive model should be focused on obtaining the highest lift. We picked the logistic regression model for Task A3.

- Create and Apply Score Dataset

- Use the same score dataset from A1.

A3.2 EXECUTION OF PLAN

- Definition of performance measures

Same as Task A1, but now we look specifically at Lift as the most important performance measure.

- Definition of approach for evaluating generated models

- The model selected was the logistic regression model.
- The model was chosen because it met the requirement of obtaining the highest lift for the top 30% of the cases.

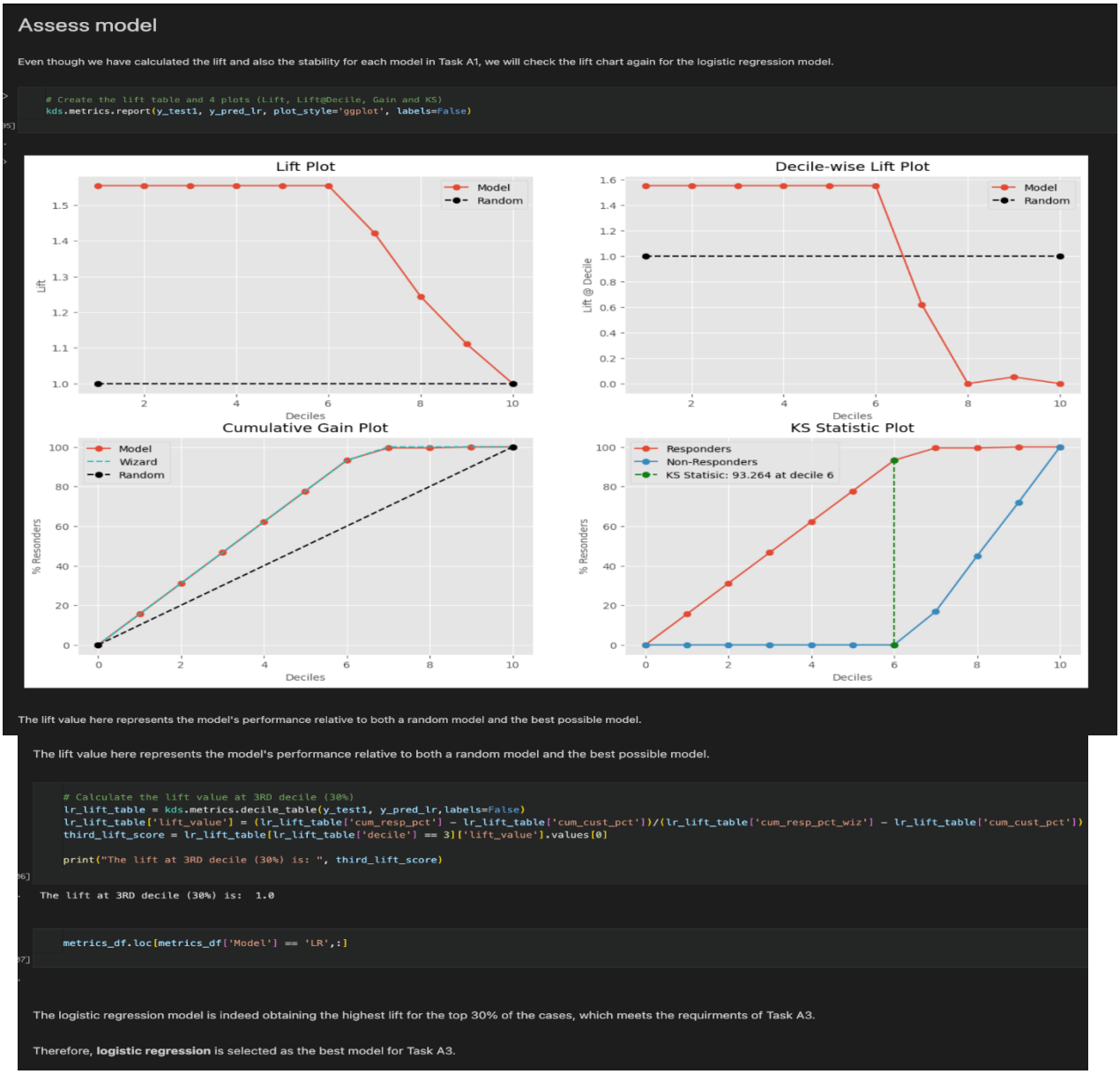


Figure 5

- Score Dataset

The same score dataset used in Task A1 was used for this task.

TASK A4

A4.1 DESCRIPTION OF PLAN

- Business Objectives Determination

- Because it is more costly to misclassify a poor loan risk as “pos” than misclassify a good loan risk as “neg,” is it possible to build a predictive model that maximizes the loans to good risk loans while also minimizing bad risk loans?
- Ability to maximize profit, minimize loss, and save money with a model

- Evaluation

- Evaluate results – Which data mining model have the best profit and loss values at the for the different outcomes?
- Review Process – describe and interpret the best model.
- Determine Next Steps – How can we tweak parameters to increase the profit and decrease the loss of the chosen model?

A4.2 EXECUTION OF PLAN

- Description of experimental approach

- The end-user experts remarked that it might also be useful to incorporate the fact that it was more costly to misclassify a person from whom a payment was really due than to misclassify a person from whom no payment was due. Ms. Imani Kuhn suggested that they should research the relevant costs.
- The result of their research was that there were an:
    - Average profit of 817.00 for correctly classifying a person for whom a payment was really due -TN
    - Average profit of 245.00 for correctly classifying a person for whom no payment was really due -TP, NoPaymentDue = ‘pos'
    - Average loss of 671.00 for misclassifying a person for whom a payment was really due - FN
    - Average loss of 465.00 for misclassifying a person for whom no payment was really due – FP
- The target event is NoPaymentDue = ‘pos’.

- A function to calculate the gross profit, average gross profit, and normalized score for profit for each of the constructed models in A1 was constructed.

- Definition of performance measures

- Gross Profit – the sum of the profit and loss:
    - gross_profit = (TP * profit_TP) + (TN * profit_TN) - (FP * loss_FP) - (FN * loss_FN)
- Average Gross Profit
    - avg_profit = gross_profit / # of cases
- Average Best Possible Profit
    - avg_bpp = (((TP + FN) * profit_TP) + ((TN + FP) * profit_TN)) / # of cases
- Normalized Score for Profit
    - normalized_score = (avg_profit - 0)/(avg_bpp - 0)

- Definition of approach for evaluating generated models

- The selected model was the logistic regression model.
- In terms of the profit metrics used, the logistic regression model was still the best model to use for Task A4.

	Model	Gross Profit	Average Profit	Normalized Score
0	SVM	130674	435.580000	0.970083
1	DT	125180	417.266667	0.929297
2	KNN	73338	244.460000	0.544438
3	XGB	127378	424.593333	0.945614
4	MNB	120050	400.166667	0.891213
5	MLP	118404	394.680000	0.878994
6	RF	125180	417.266667	0.929297
7	LR	133788	445.960000	0.993200

Figure 6

- Score data set

Same score dataset used in Task A1 was used for Task A4.

TASK A5

A5.1 DESCRIPTION OF PLAN

- Business Objectives Determination

Can a student's characteristics and history, provided during and at the time of the application process, be utilized to assess his/her risk level with accuracy with the highest lift?

- Data Understanding

- Collect Initial Data
  - Use the ordinal and interval data types only.
    - "Longest_Absence_From_School" , "LongTermPlanningScore", "PowerOrientedScore", "CommunityOrientedScore", "Family_Income_normalized"
    - "Parent_Education_Level", "Credit_Score_Raw", "Credit_Score_AgeAdj", "Loan_Amount"
    - "HS_Math", "HS_Science", "HS_English"

- Use the original Loan_Amount data, do not break it into two groups as we did for the other tasks.

- ## Data Preparation

- Select Data
  - Create a dataset that needs to be normalized
  - Need to deal with these columns with different scales.
- Format Data
  - Use some sort of scaler to normalize the data.

- ## Data Mining

- Use a visualizer and create functions to get clustering labels for segmentations.
- Assess Model –
  - The end users defined a useful segmentation would be one in which for at least 2 of the 3 groups: a) the proportion of students with NoPaymentDue = 'neg' was very different from the corresponding overall proportion for the entire dataset; and b) the difference in the proportions of students with NoPaymentDue = 'neg' for the 2 groups was statistically significant.
  - Compare the segmentations to see which one is better or more suited to the prompt. Then evaluate the validity of those clusters.

- ## Evaluation

- Which segmentation describes 2 out of the 3 groups as best as the prompt describes?
- Interpret the segmentations
- Score dataset

A5.2 EXECUTION OF PLAN

- ## Description of Data Preparation

- Select Data
  - Create a data set that needed to be normalized.
  - The data set includes the target event, as well as the columns listed in the A5 planning.
  - Add the 'Loan_Amount' column to the data set.
- Format Data
  - To normalize the data, use the MinMaxScaler to scale the data set.
  - Drop the 'Family_Income_Normalized' column because it is already normalized.
  - Get the columns names as a list
  - Scale the columns with the MinMaxScaler
  - Add the 'Family_Income_Normalized' column back to the dataset

- ## Description of experimental approach

- The project team desired a model that created 3 clusters of students based on the interval and ordinal variables.
- Build Models – 2 cluster segmentations:
    - LKER_3(lloyd, k-means++, Euclidean, Range, 3 clusters)
    - MRCR_3(MacQueen Random chi-square range_feature 3 clusters)
- Hyperparameter tuning – Hyperparameters listed above.

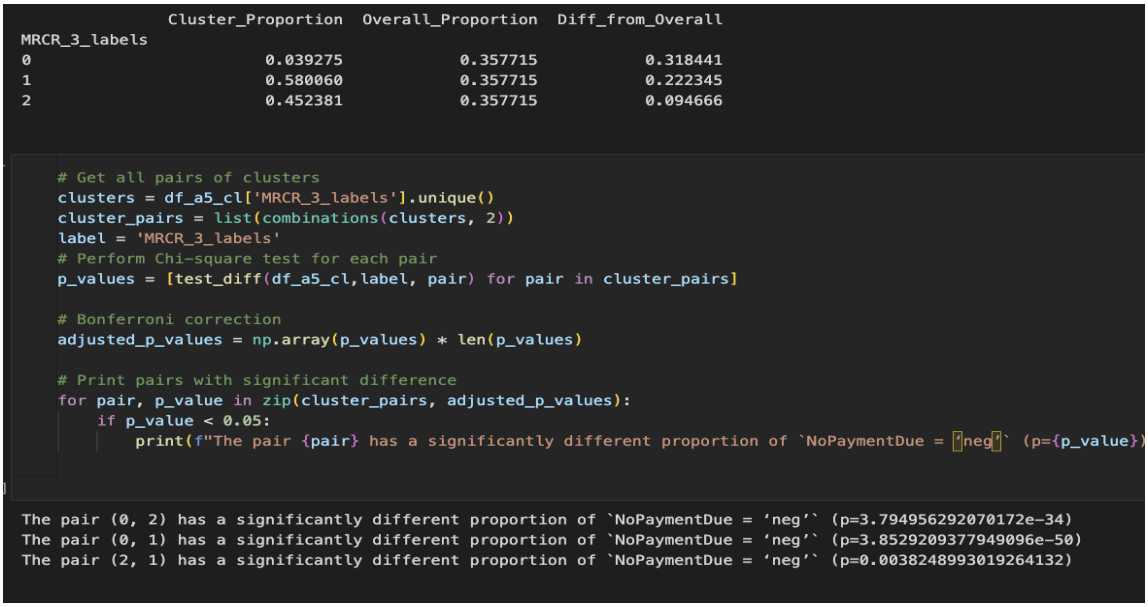
Definition of performance measures

- The end users defined a useful segmentation would be one in which for at least 2 of the 3 groups:
    - a) the proportion of students with NoPaymentDue = 'neg' was very different from the corresponding overall proportion for the entire dataset.
    - b) the difference in the proportions of students with NoPaymentDue = 'neg' for the 2 groups was statistically significant.

Definition of approach for evaluating generated models

- Cluster models

Figure 7



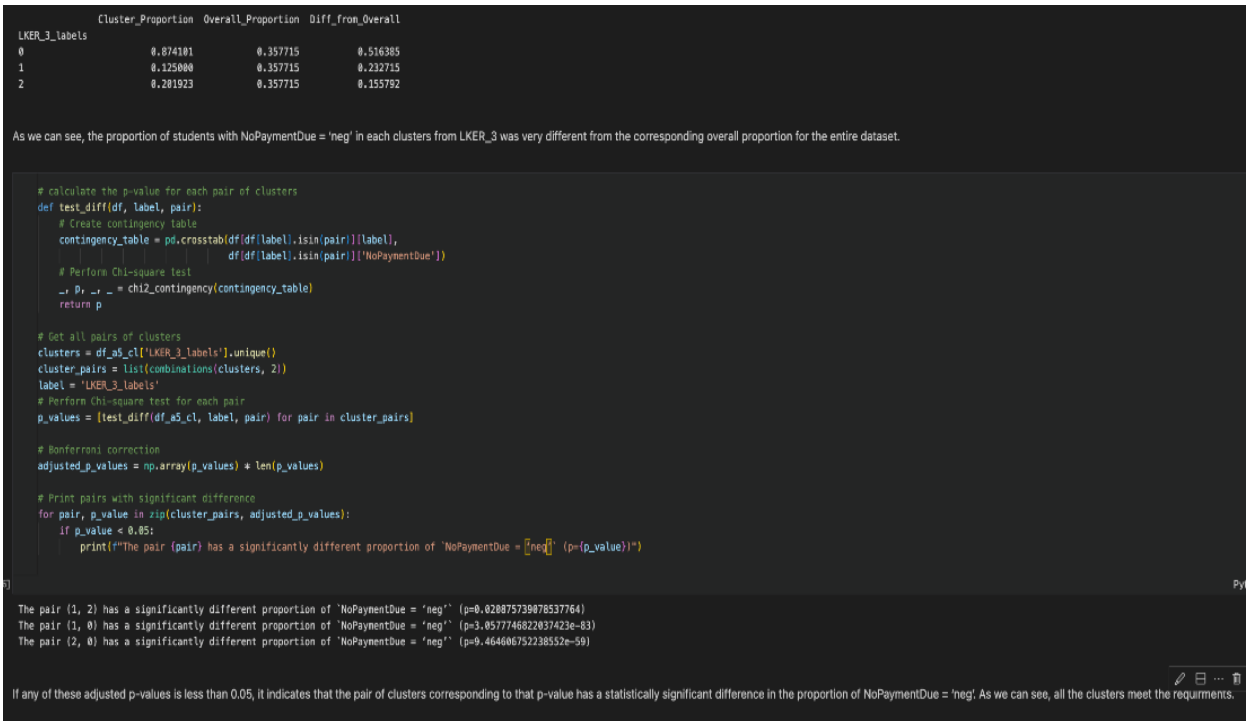
MRCR_3_labels	Cluster_Proportion	Overall_Proportion	Diff_from_Overall
0	0.039275	0.357715	0.318441
1	0.580060	0.357715	0.222345
2	0.452381	0.357715	0.094666

```
# Get all pairs of clusters
clusters = df_a5_cl['MRCR_3_labels'].unique()
cluster_pairs = list(combinations(clusters, 2))
label = 'MRCR_3_labels'
# Perform Chi-square test for each pair
p_values = [test_diff(df_a5_cl,label, pair) for pair in cluster_pairs]

# Bonferroni correction
adjusted_p_values = np.array(p_values) * len(p_values)

# Print pairs with significant difference
for pair, p_value in zip(cluster_pairs, adjusted_p_values):
    if p_value < 0.05:
        print(f"The pair {pair} has a significantly different proportion of `NoPaymentDue = 'neg'` (p={p_value})
```

```
The pair (0, 2) has a significantly different proportion of `NoPaymentDue = 'neg'` (p=3.794956292070172e-34)
The pair (0, 1) has a significantly different proportion of `NoPaymentDue = 'neg'` (p=3.8529209377949096e-50)
The pair (2, 1) has a significantly different proportion of `NoPaymentDue = 'neg'` (p=0.0038248993019264132)
```

Figure 8

- Cluster Proportions



LKER_3_labels	Cluster_Proportion	Overall_Proportion	Diff_from_Overall
0	0.874101	0.357715	0.516385
1	0.125000	0.357715	0.232715
2	0.201923	0.357715	0.155792

As we can see, the proportion of students with NoPaymentDue = 'neg' in each clusters from LKER_3 was very different from the corresponding overall proportion for the entire dataset.

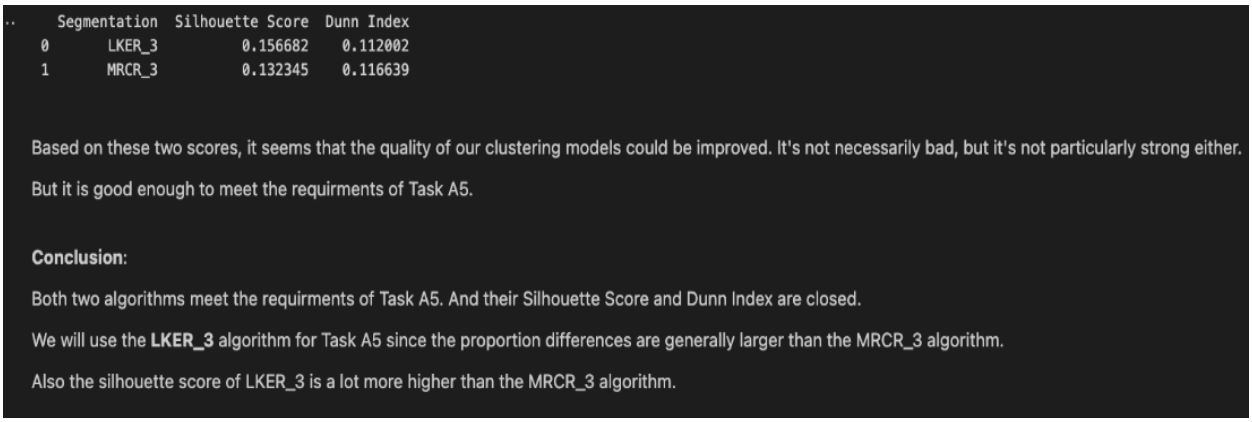
```
# calculate the p-value for each pair of clusters
def test_diff(df, label, pair):
    # Create contingency table
    contingency_table = pd.crosstab(df[df[label].isin(pair)][label],
                                    df[df[label].isin(pair)]['NoPaymentDue'])
    # Perform Chi-square test
    _, p, _, _ = chi2_contingency(contingency_table)
    return p

# Get all pairs of clusters
clusters = df_a5_cl['LKER_3_labels'].unique()
cluster_pairs = list(combinations(clusters, 2))
label = 'LKER_3_labels'
# Perform Chi-square test for each pair
p_values = [test_diff(df_a5_cl, label, pair) for pair in cluster_pairs]

# Bonferroni correction
adjusted_p_values = np.array(p_values) * len(p_values)

# Print pairs with significant difference
for pair, p_value in zip(cluster_pairs, adjusted_p_values):
    if p_value < 0.05:
        print(f"The pair {pair} has a significantly different proportion of `NoPaymentDue = 'neg'` (p={p_value})")
```

```
The pair (1, 2) has a significantly different proportion of `NoPaymentDue = 'neg'` (p=0.020875739078537764)
The pair (1, 0) has a significantly different proportion of `NoPaymentDue = 'neg'` (p=3.0577746822037423e-83)
The pair (2, 0) has a significantly different proportion of `NoPaymentDue = 'neg'` (p=9.464606752238552e-59)
```

If any of these adjusted p-values is less than 0.05, it indicates that the pair of clusters corresponding to that p-value has a statistically significant difference in the proportion of NoPaymentDue = 'neg'. As we can see, all the clusters meet the requirments.

Figure 9

- Conclusion



	Segmentation	Silhouette Score	Dunn Index
0	LKER_3	0.156682	0.112002
1	MRCR_3	0.132345	0.116639

Based on these two scores, it seems that the quality of our clustering models could be improved. It's not necessarily bad, but it's not particularly strong either.

But it is good enough to meet the requirments of Task A5.

Conclusion:

Both two algorithms meet the requirments of Task A5. And their Silhouette Score and Dunn Index are closed.

We will use the **LKER_3** algorithm for Task A5 since the proportion differences are generally larger than the MRCR_3 algorithm.

Also the silhouette score of LKER_3 is a lot more higher than the MRCR_3 algorithm.

Figure 10

- Score data set
  - A score data set with 35 new cases was generated.
  - Clustering model was then applied to the score data set.

Part B

Task B1: Best Model Recommendation Using MSE

B1.1 DESCRIPTION OF PLAN

- Data Understanding

Collect Initial Data

As a single .csv file, the DMABASE dataset was available on a shared drive: https://cgu.instructure.com/courses/11475/files/folder/Student%20Documents/Final%20Project%20Documents/Datasets. The data source had a total of 23 variables which is outlined in Table 7.

Variable Name	Role	Measurement	Type	Description
NAME	id	nominal	char	Player's Name
TEAM	input	nominal	char	Team at the end of 1986
NO_ATBAT	input	interval	num	Times at Bat in 1986
NO_HITS	input	interval	num	Hits in 1986
NO_HOME	input	interval	num	Home Runs in 1986
NO_RUNS	input	interval	num	Runs in 1986
NO_RBI	target	interval	num	RBIs in 1986
NO_BB	input	interval	num	Walks in 1986

YR_MAJOR	input	interval	num	Years in the Major Leagues
CR_ATBAT	input	interval	num	Career times at bat
CR_HITS	input	interval	num	Career Hits
CR_HOME	input	interval	num	Career Home Runs
CR_RUNS	input	interval	num	Career Runs
CR_RBI	input	interval	num	Career RBIs
CR_BB	input	interval	num	Career Walks
LEAGUE	input	binary	char	League at the end of 1986
DIVISION	input	binary	char	Division at the end of 1986
POSITION	input	nominal	char	Position(s) in 1986
NO_OUTS	input	interval	num	Put Outs in 1986
NO_ASSTS	input	interval	num	Assists in 1986
NO_ERROR	input	interval	num	Errors in 1986
SALARY	input	interval	num	1987 Salary in $ Thousands
LOGSALAR	input	interval	num	Log Salary

Table 7: Data Dictionary for the DMABASE dataset

The file was uploaded to the Project Team's GitHub Repository and Google Drive where the data can be read into a python environment and notebook.

Describe Data:

With 322 records and 23 variables (including logsalar which is our target variable), the data source was provided as a .csv file. There are 5 categorical and 18 numerical variables. Upon reviewing the file, the data provided does meet the predictive modeling requirements outlined in the project description. The following tables display our initial data collection report and a descriptive statistics summary of the numerical variables (Table 8 and Table 9).

Column Name	Data Type	Missing Values	Missing Values %	Unique Values	Unique Values %	Min	Max	STD
name	object	0	0	322	100	NaN	NaN	NaN
team	object	0	0	24	7.45	NaN	NaN	NaN
no_atbat	int64	0	0	246	76.4	127	687	143.595835
no_hits	int64	0	0	136	42.24	31	238	44.179509
no_home	int64	0	0	37	11.49	0	40	8.69877
no_runs	int64	0	0	91	28.26	12	130	25.057366
no_rbi	int64	0	0	100	31.06	8	121	25.501162
no_bb	int64	0	0	85	26.4	3	105	21.095941
yr_major	int64	0	0	22	6.83	1	24	4.969707
cr_atbat	int64	0	0	315	97.83	166	14053	2328.479167
cr_hits	int64	0	0	287	89.13	34	4256	654.787619
cr_home	int64	0	0	151	46.89	0	548	90.065127
cr_runs	int64	0	0	262	81.37	18	2165	336.425038

cr_rbi	int64	0	0	264	81.99	9	1659	338.790345
cr_bb	int64	0	0	248	77.02	8	1566	273.625372
league	object	0	0	2	0.62	NaN	NaN	NaN
division	object	0	0	2	0.62	NaN	NaN	NaN
position	object	0	0	25	7.76	NaN	NaN	NaN
no_outs	int64	0	0	234	72.67	0	1378	280.656673
no_assts	int64	0	0	161	50	0	492	136.852454
no_error	int64	0	0	29	9.01	0	32	6.368359
salary	float64	59	18.32	150	46.58	67.5	2460	451.118681
logsalar	float64	59	18.32	150	46.58	4.212128	7.807917	0.889192

Table 8: Initial Data Collection Report for the DMABASE dataset.

Upon our initial review of the dataset, here is a list of initial problems, questions, and thoughts:

- NAME is a unique id, so we will most likely have to drop this column because it is useless for modeling.
- Since we have both a salary and logsalar (which is a log of the salary column) variables, we can drop salary and just keep the logsalar variable.
- 18% of the logsalar variable is missing, so these values will need to be imputed.
- Does career stats include 1986 stats as well? Though we are lacking business understanding, our Team Lead mentioned that the 1986 statistics and the career statistics are separated; thus, 1986 data are not included in career, and they are not supposed to be combined. Also, the dependent variables do not have data from 1987.
- What derived data can we use? As directed by our Team Lead, baseball statistics such as OB% and Bating AVG could be useful derived variables to use if data transformations are needed. It was also confirmed that no additional statistics need to be added to the data set.
- Because we have data from 1986 and overall career, we can expect some highly correlated variables.
- Some variables have some high values, so we might have to consider binning or scaling them or defaulting to ensemble models which are not as sensitive to outliers and skewed data.

To understand how the dependent variables are impacting logsalar, we did a quick correlation analysis which is shown here (figure #).

Explore Data:

Our data exploration plan includes reviewing the metadata, looking at the descriptive statistics of the variables, distribution, correlation analysis, and creating visualizations using histograms and bar charts to really understand each variable. For categorical variables, we will also use visualizations and value counts to see and understand each categorical values within the variables.

Verify Data:

To verify the data, we plan on taking a closer look at all the variables not only to explore the data but to check for missing values, errors, duplicate records, high values, outliers, and the variables relationship with each other logsalar. We also plan on checking for useless variables such as unique identifiers, variables with the same value, or records that only account for 1 value in a column. For any data quality issues, we plan on addressing them with our Team Lead. However, since the business understanding

was not available to us, we may have to refer to 3rd party baseball sources such as the https://www.baseball-almanac.com/, https://en.wikipedia.org/wiki/Baseball, https://www.baseball-reference.com/, and maybe others.

- ## Data Preparation

Clean Data

- For variables missing any data, we will have to employ the most appropriate imputation method based on the variable and exploration of that variable. Specifically for this case, we will have to do this on the logsalar variable. Since baseball salaries are dependent on player performance, doing a tree imputation on the entire dataset for logsalar makes the most sense.
- Drop variables unique identifiers such as the name column and duplicated variables such as the salary column.

Because we only have 1 data source, we will not need to integrate multiple data sources together. However, if our models do not perform well, we may have to resort to replacing categorical variables with statistical data from the baseball website. We plan on using this as a last resort.

Format Data

- Review and explore derived variable options through regrouping or data transformation of categorical variables as well as standardizing any inconsistencies within columns such as variations of spelling. Also, we plan to address any data anomalies such as the odd abbreviations in the position column.
- Since we are required to do some clustering, we will have to normalize the data.
- Since we are required to do a decision tree classification model, we will have to do a data transformation on the logsalar variable from interval to categorical ordinal variable.
- Ensure that all variables are the correct data type.

Construct Data

- Use one-hot encoding on categorical data (team, position, division, league).
- If too many variables, we will use label encoding.
- Use ordinal encoding for logsalar for B3.

- ## Data Mining

Modeling

- Plan for modeling techniques that are not sensitive to outliers and skewed data.
- Test Design: 70% training and 30% test split
- Because of the outliers and skewed data, we are focusing on following ensemble models because they are not sensitive to this type of data:
    - Regression Tree
    - Random Forest Regression

    - Gradient Boosting Regression
    - AdaBoost Regression
- Hyperparameter tuning: GridSearchCV and RandomSeach techniques
- Assessment Measures: MSE, R^2, and Stability
- Post Pruning: If needed, post pruning will be done on the Regression Tree

Upon completing the data understanding, here is our overall data preparation plan:

1. Because it is unique variables, drop the player's name.
2. Do some data cleansing with the position's variable; for those odd position abbreviations, we will have to verify those abbreviations, and perhaps do some position groupings.
3. If we consider a clustering model, we'll have to normalize the entire dataset.
4. We can drop the salary variable since we have logsalar.
5. We will have to do input the logsalar variable using random forest imputation.

In addition, we will keep in the following:

- The Correlations Analysis shows some high correlated variables. This might impact modeling; thus, if we are seeing poor performance during modeling, we may need to address this.
- If we use a tree-based model, we won't have to worry about normalizing the data and encoding the categorical variables.
- If we use a clustering model, we'll have to normalize the data and encode the categorical variables.
- Since the records are small, we may use PCA to reduce the dimensionality of the data.
- If the models are overfitting, we may have to resort to label encoding instead of one-hot encoding and do some post-pruning.

- Evaluation

Evaluate the results of each model and choose the best model with the lowest mean squared error (MSE). We will also factor in the stability and R^2 as well.

B1.2 EXECUTION OF PLAN

- Data Understanding

Collect Initial Data

The DMABASE dataset was downloaded from the shared drive, read into a python environment as a Pandas data frame, and inspected the first 5 records of the dataset.

Describe Data

First, we examined the metadata of the dataset using a function (see Table 8) to understand each variable from a high-level perspective and see the descriptive statistics of all the numerical variables.

From the function, we were able to see the data types, identify variables with missing data, percentage of missing data, number of unique values, and percentage of unique values.

Explore Data

An exploratory data analysis was conducted on the entire dataset. First, we conducted a correlations analysis of the numerical variables (Figure #).

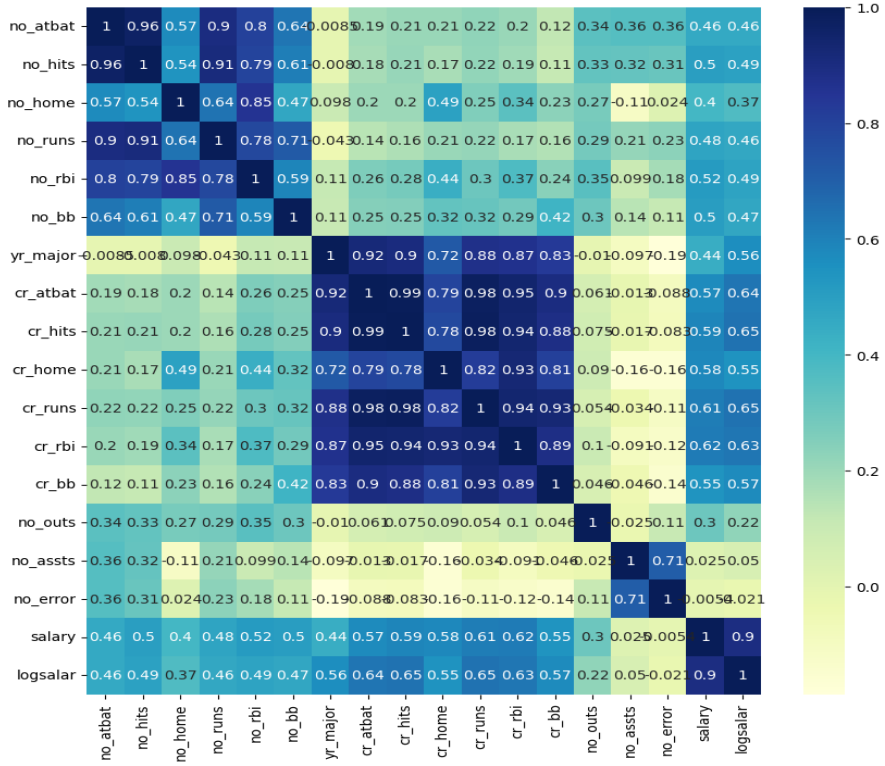


Figure: Correlation Matrix/Heatmap of the interval variables of the DMABASE dataset.

Based Figure #, many of the variables have positive correlation with logsalar. Also, instead of a correlation with career and 1986 statistics, there are correlations within both time spans. This next section provides an overall summary:

- no_hits is highly correlated with no_atbats, no_runs, and no_rbi
- no_home is highly correlated with no_rbi
- no_runs is highly correlated with no_atbats, no_rbi, and no_bb
- no_rbi is highly correlated with no_hits, no_home
- Career stats and yr_major are all highly correlated with each other
- Salary and logsalar are highly correlated with each other, but Salary will be dropped since we have the logsalar, so we won't have to worry about this.

With 1986 and career statistics, here are some possible data transformations to keep in mind:

- **On-base percentage (OBP)**: measures a player's ability to get on base (excluding fielding errors or other mistakes by the defensive team).

OBP = (Hits + Walks + Hit-by-Pitches) / (At-Bats + Walks + Hit-by-Pitches + Sacrifice Flies)

- Batting Average = the number of hits divided the total number At-Bats (does not include walks).

Hits / At-Bats

Next, we explored every variable using a data exploration function. With the help of this function, we were able to visualize the basic statistics using bar charts, box and whisker plots, and histograms so that we can visualize the distribution, identify any data skews, identify outliers, see the mean, and standard deviation.

Verify Data

Upon verifying the data, we were able to leverage the unique and total counts of each variable as well as compare the data from the dataset to historical baseball data available on the website mentioned above.

• Data Preparation

Clean Data

- Drop useless variables
  - Salary: we do not need this because we have logsalary variable which is the log of the salary column.
  - Name: this is a unique identifier; thus, useless for predictive modeling.
- Missing values
  - Logsalar: with this column missing 18% of the data, we imputed these values using MissForest library within missingpy. We decided to use this method because salary is based on baseball performance; thus, we imputed the variable based on the entire dataset. We will keep in mind that imputing the target variable with MissForest, a tree imputation method, might introduce bias during modeling.

Format Data

- Derive new variables
  - Using team, league, and division variables, we will concatenate them to create a new column called "team_league_division." This will allow us not only separate those cities that have multiple team but also help us encode this variable based on their 1986 ranking. Once this new variable was created, we were able to drop league and division.

Construct Data

- Group Data
  - Position: using the basic baseball positions, looked up the unknown/odd abbreviations and determined that those players played multiple positions. However, the abbreviation in the dataset was a combination of their top two positions with the most outs. Those with "u" in the abbreviation had high batting statistics. Thus, we regrouped all these

positions using the top position. For example, if a player had a 2S abbreviation, we know that this player had the most outs playing 2nd base and short stop for that team in 1986. Because 2 is first, we know this was the players top position was 2nd base; thus, this player was grouped into the "2B" category.

- Position/Team: we used one-hot encoding for position and team

Experimentation

As mentioned in the Format Data section, we created a derived variable to setup label encoding. Upon initially doing one-hot encoding, we had very poor performing models. They were all overfitting. In light of this, we switched from one-hot encoding to label encoding for position and team, and created our new variable team_league_division. For position, we encoded using transform function, and for team_league_division, we encoded based on the team's 1986 ranking. Based on some baseball research, baseball salaries are based on player performance, so if a player played in the playoffs or was on a team that won the World Series, this will impact these players salaries for the following year.

• Data Mining

Train/Test/Split

70%/30% split was conducted on the dataset into training and testing respectively.

Modeling

- Regression Tree, Random Forest Regression, Gradient Boosting Regression, and AdaBoost Regression
- Hyperparameter tuning: RandomSeach techniques

It should be noted here that initial modeling was conducted with the one-hot encoded dataset. Since the models were overfitting, we reencoded the dataset using label encoding and did and ran the new dataset with our existing models. Our models performed much better the 2nd time around.

• Evaluation

Definition of Performance Measures

Measure	Description	Definition of Value Function
Mean Squared Error	Measures the average squared difference between the estimated values and the actual values.	$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$ MSE = mean squared error n = number of data points Y_i = observed values $\hat{Y}_i$ = predicted values

R^2	Proportion of the dependent variable variance that's explained by the independent variable in a regression model.	$$R^2 = 1 - \frac{RSS}{TSS}$$ R^2 = coefficient of determination RSS = sum of squares of residuals TSS = total sum of squares
Stability	Stability of the coefficients relative to the data; calculated between 0-1. The higher the score, the higher the stability.	MIN (training R^2/testing R^2, testing R^2/training R^2)

Results

	Model	MSE	R2_train	R2_test	Stability
0	regression tree	0.166957	0.866403	0.781180	0.901636
1	regression tree post-pruning	0.165275	0.863790	0.783385	0.906915
2	random forest	0.157730	0.921126	0.793273	0.861200
3	gradient boosting	0.154335	0.999818	0.797722	0.797868
4	ada boost	0.173139	0.919465	0.773078	0.840790

Because B1's requirement is to select the model with the lowest MSE, the gradient boosting model would be best performing in this case because it has the lowest MSE out of all 5 models. However, this model is also the least stable. Thus, while we feel that the gradient boosting is the best based on the B1 requirements, we would also keep in mind our post pruned regression tree and random forest models. Even though these models have a slightly higher MSE, these models are much more stable in the long run. We would also recommend adding additional player and team statistics to further improve the performance.

TASK B2

B2.1 DESCRIPTION OF PLAN

- Business Objectives Determination
  - Provide different cluster segmentations where the maximum number of clusters should be less or equal to the number of rules in your choice of the most appropriate decision tree for Task B1.
- Data Understanding
  - Analyze the metadata of the raw data to see what needs to be done to make good clusters for the task.
- Data Preparation
  - Select Data
    - Drop the useless variables.

        - Drop the target variable.
    - Format Data
        - Use binary encoding on the league and division variables.
        - Use one-hot encoding on the position and team variables.
        - Normalize the data
- Data Mining
    - Use the Elbow Score to determine the optimal number of KMeans clustering
    - Determine the amount of clusters, counts, and labels in each cluster segmentation.
    - Which segmentation has the desired number of clusters?
- Evaluation
    - See which segmentations have the desired range of clusters.
    - Compare their performances through the Silhouette score and Dunn Index.

B2.2 EXECUTION OF PLAN

- Data Preparation
    - Select Data
        - Drop the player's name and salary variables because they are of no use.
        - Drop the target variable 'logsalar', not wanted for the creation of the clusters.
    - Format Data
        - Use binary encoding on the league and division variables.
            - Binary encode the league variable, American:0, National:1
            - Binary encode the division variable, East:0, West:1
        - Use one-hot encoding on the position and team variables.
            - Create a list of the basic positions
            - Map the special positions to their corresponding basic positions
                - # CS is a special case, it actually means CF and 3B based on the research
            - If the position is a special one, replace it with the first corresponding basic position
            - Generate one-hot encoding for the basic positions
            - Manually adjust the one-hot encoding for the special positions
            - Drop the original 'positions' column
        - Normalize the data with the MinMaxScaler
            - Concatenate scaled and unscaled features
- Description of experimental approach
    - The project team desired a model that created 3 clusters of students based on the interval and ordinal variables.
    - Build Models – 2 cluster segmentations:
        - LKER_3(lloyd, k-means++, Euclidean, Range, 3 clusters)
        - MRCR_3(MacQueen Random chi-square range_feature 3 clusters)
    - Hyperparameter tuning – Hyperparameters listed above.
- Definition of performance measures
    - Number of clusters in segmentation is less than or equal to the amount of rules in the DT selected in B1.

- Silhouette Score = (b-a)/max(a,b)
- Dunn Index: (minimum inter-cluster distance) /(the maximum cluster size0

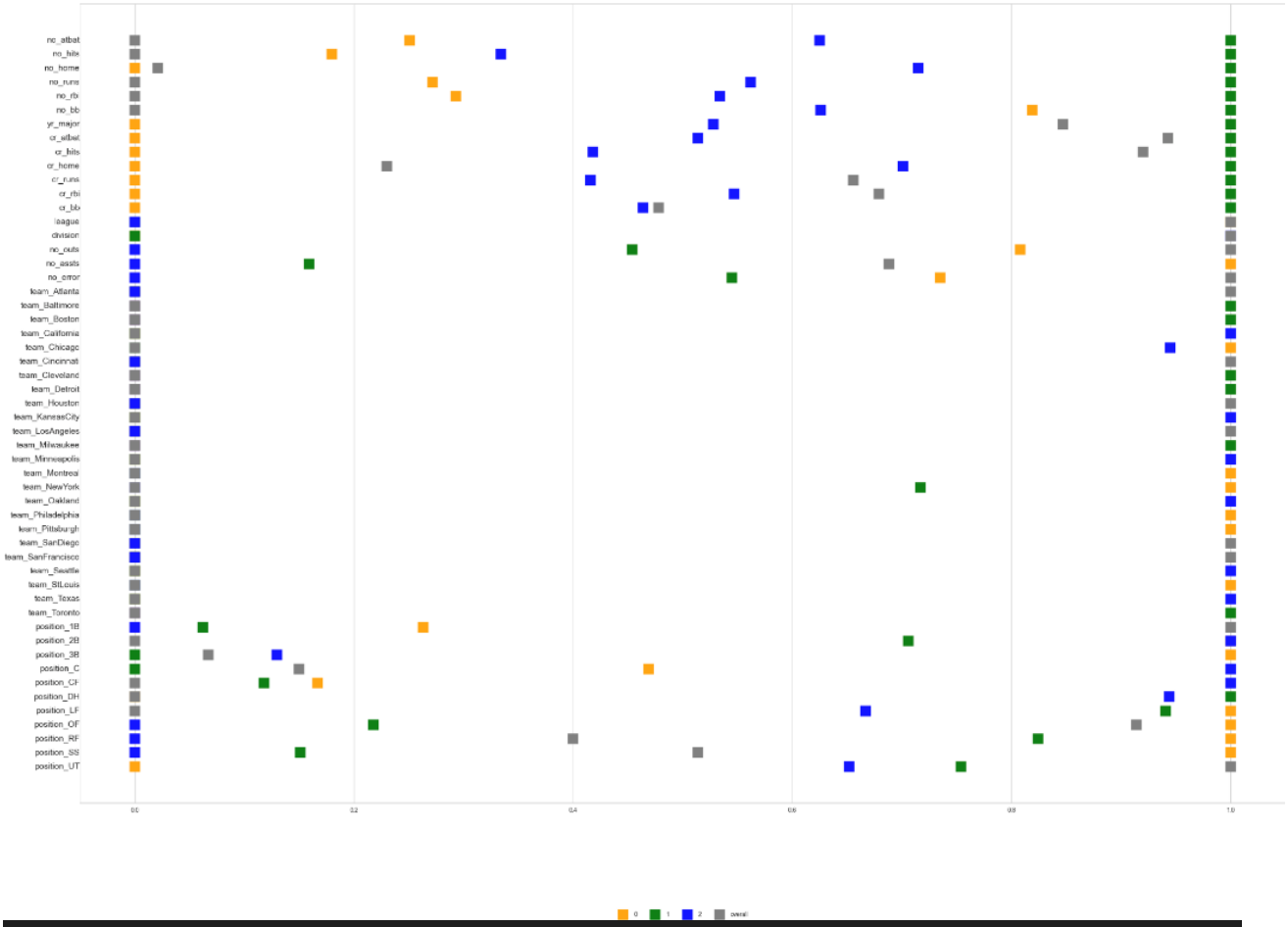
- Definition of approach for evaluating generated models

- The clusters generated were
  - LKER_4 (lloyd, k-means++, Euclidean, Range, 4)
  - ERER_3 (elkan, random, Euclidean, Range, 3)
  - MRMR_4 (MacQueen, Random, manhattan, Range, 4)
- Segmentations were assessed and compared. The LKER_4 segmentation had the highest Silhouette score and Dunn Index out of all three segmentations.

	Segmentation	Silhouette Score	Dunn Index
0	LKER_4	0.107061	0.371995
1	EKER_3	0.088222	0.347975
2	MRMR_4	0.067521	0.271400

LKER_4 has the best performance among the three clustering models. It has the highest silhouette score and Dunn index.

- The normalized mean of the LKER_4 segmentation was plotted and analyzed.



It seems that there are indeed differences between the clusters for many features (for example, 'no_atbat', 'no_hits', etc.). This suggests that the clustering is capturing some structure in the data. However, there are also features that do not show clear differences between the clusters (for example, 'team_Atlanta', 'team_Baltimore', etc.), indicating that these features are not contributing much to the clustering.

Conclusion:

LKER_4 is the best clustering model for this dataset. It has the highest silhouette score and Dunn index. The clusters are well separated and the cluster means plot shows that there are indeed differences between the clusters for many features. This suggests that the clustering is capturing some structure in the data.

TASK B3

B3.1 DESCRIPTION OF PLAN

- Business Objectives Determination
  - Generate the best Decision Tree model that partitions the continuous variable LOGSALAR into three intervals: (0, Mean –0.25*StdDev), (Mean – 0.25*StdDev, Mean + 0.25*StdDev], (Mean + 0.25*StdDev, +infinity].
- Data Understanding
  - Use the imputed data set from Task B1.
- Data Preparation
  - Format Data

- Calculate the mean and standard deviation of the logsalar variable.
    - Partition the logsalar variable into three variables into the three intervals described in the prompt.

- Data Mining
    - Split data into training and testing groups – 80% training and 20% testing.
    - Generate decision tree models that use different scoring methods.

- Evaluation
- Evaluate accuracy and determined performance measures such as recall, precision, f1, etc.
- See which scoring method yields the highest accuracy across the models.

B3.2 EXECUTION OF PLAN

- Data Preparation
- Select Data
    - Drop the player's name and salary variables because they are of no use.
    - Drop the target variable 'logsalar', not wanted for the creation of the clusters.
- Format Data
    - Calculate the mean and standard deviation of the logsalar variable.
    - Partition the logsalar variable into three variables into the three intervals described in the prompt.

- Description of experimental approach
- The project team desired a high-quality decision tree model that partitioned the 'logsalar' column into three intervals.
- Build Models – 4 Decision Tree Models:
    - Dt3_acc({'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf': 7, 'min_samples_split': 20})
        - Scoring method: 'accuracy'
    - Dt3_bal_acc ('criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2})
        - Scoring method: 'balanced_accuracy'
        - Dt3_f1('criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 10, 'min_samples_split': 2})
            - Scoring method: 'f1_weighted'
        - Dt3_roc ({'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 10, 'min_samples_split': 2})
            - Scoring method: 'roc_auc_ovr'

- Definition of performance measures
- 1. Accuracy: Ratio of correct predictions to total predictions. Useful when classes are nearly balanced.
- 2. Precision (Macro): Measures how many correctly predicted positive instances out of all predicted positive instances. Macro averages this across all classes.
- 3. Recall (Macro): Measures how many correctly predicted positive instances out of all actual positive instances. Macro averages this across all classes.

- 4. F1 Score (Macro): Harmonic mean of Precision and Recall providing a balance between the two. Macro averages this across all classes.
- 5. Balanced Accuracy: Average of recall for each class, useful for imbalanced datasets.
- 6. AUC-ROC (Micro): Measures the model's ability to distinguish between classes. Micro treats the multiclass problem as a binary one, considering each prediction in a one-vs-all manner.

- Definition of approach for evaluating generated models
  - The performance measures described above were how the models were evaluated. The model with the highest-performing metrics was selected.
  - The dt3_acc model was selected. This model had the highest accuracy of the models, as well as the highest recall, f1, balanced accuracy, and AUC_ROC.

	Model	Accuracy	Precision	Recall	F1	Balanced Accuracy	AUC_ROC
0	dt3_acc	0.892300	0.865900	0.876400	0.870500	0.876400	0.919200
1	dt3_bal_acc	0.861500	0.830000	0.834600	0.831200	0.834600	0.896200
2	dt3_f1	0.876900	0.842700	0.864900	0.849600	0.864900	0.907700
3	dt3_roc	0.892300	0.871000	0.838800	0.850700	0.838800	0.919200

The dt3_acc which {'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf': 7, 'min_samples_split': 20} is the best decision tree model for this task B3.

Because it has the highest accuracy score while having the relatively high values of other metrics.