|  |  |
| --- | --- |
| **DATA 430 Technical Report Assignment 2: Bayesian Classification** | **Shawnequa King** |
| **Bayesian Classification** | |
| **URL to dataset: https://www.kaggle.com/datasets/ruthgn/bank-marketing-data-set?resource=download** | |

This template should be used in conjunction with the assignment instructions. The size of the text area below will expand to the length of your response; the area should not be interpreted as a required or suggested length of response. Responses within the text area should be single spaced with Times New Roman 12pt font. The body of the document will likely be 6-9 pages, not including the Appendix; length may vary depending on specifics of the analysis and the dataset. As needed, APA format in-text citations should be included, along with a full references list at the end of the document.

|  |
| --- |
| **Overview** |
| **Problem Domain**: give some background and context about the problem domain (application area). For instance, if you are doing the analysis for predicting heart disease, provide some context about the disease and include some interesting statistics about it. Also, discuss how the method is relevant for the chosen problem. |
| The Bank marketing dataset has 41188 examples with 20 inputs and 1 output variable. This data is from Portuguese Banking institution. It has numerical as well as categorical attributes and response attribute y denotes client subscribed to term deposit or not (yes or no). The goal is to build models that can predict if client will subscribe to term deposit or not. Since response variable is binary, different classification models will be used incrementally till it gives model with best accuracy.  Insider Intelligence predicts banks will spend $13.54 billion on digital advertising in 2022 and $15.08 billion in 2023. (Reyes, 2022). As a result, **banking marketers may see a general downward trend in pricing in the next few quarters** because some deals may be renegotiated, which seldom happens on national TV. Broadcast TV (big networks like ABC, CBS and NBC) will be in a stronger negotiating position than cable (Reyes, 2022). |
| **Objective**: clearly state the objective of the analysis in relation to the kind of algorithm you are employing. Use specific language as to what question(s) you are trying to answer using the specific analysis/modeling type. |
| The dataset is downloaded from UCI Machine Learning Repository and is related to direct marketing campaigns of a Portuguese Banking institution. These campaigns were based on phone calls. Often, more than one calls were done to the same client to access if their product “term deposit” will be subscribed (yes) or not subscribed(no). This dataset is available at [**https://www.kaggle.com/datasets/ruthgn/bank-marketing-data-set?resource=download**](https://www.kaggle.com/datasets/ruthgn/bank-marketing-data-set?resource=download)  There were 4 datasets in it from which bank-additional-full.csv is used that has all examples (41188) and 20 inputs ordered by date (from May 2008 to November 2010). There are 20 input variables and 1 output variable (desired target). The dataset had different types of client data like age, job, martial, education, default, housing, loan, contact, month, day\_of\_week, duration, campaign, pdays, previous, poutcome, em.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr. Employed and one output variable y that denotes if client subscribed to term deposit or not. These dataset attributes denote customer data, socio-economic data, telemarketing data and some other data. Some attributes are numerical, and some are categorical. The dataset was loaded in R Studio and checked for any missing values using is.na function and found that it didn’t have any missing values. So, we have a clean dataset. |
| **Analysis** |
| **Exploratory Analysis**: describe the data including the source, the collection method, and variables. Perform exploratory analysis. Also, select few key variables (including the target variable for supervised learning) and study their distributions using plots such as histograms, box plot, bar chart, etc. |
| This dataset has different types of client data which are listed below:   1. age – Client Age- (numeric) 2. job – Type of Job - (categorical) ('admin.','blue-collar','entrepreneur','housemaid','management','retired','self- employed','services','student','technician','unemployed','unknown') 3. marital - Client’s marital status - (categorical) (divorced, married, single, unknown, note divorced means divorced or widowed) 4. education - Client’s education - (categorical) (basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree, unknown) 5. default - has credit in default? - (categorical) (no, yes, unknown) 6. housing - Has housing loan? - (categorical) (no, yes, unknown) 7. loan - has personal loan? - (categorical) (no, yes, unknown’) 8. contact – last contact month of year - (categorical) (cellular, telephone) 9. month - Month of last contact with client - (categorical) (January - December) 10. day\_of\_week - last contact day of the week - (categorical) (Monday - Friday) 11. duration - last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). 12. campaign: number of contacts performed during this campaign and for this client (numeric) 13. pdays - number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means clients were not previously contacted) 14. previous - Number of client contacts performed before this campaign - (numeric) 15. poutcome - outcome of the previous marketing campaign - (categorical) (failure, nonexistent, success) 16. emp.var.rate - Quarterly employment variation rate - (numeric) 17. cons.price.idx - Monthly consumer price index - (numeric) 18. cons.conf.idx - Monthly consumer confidence index - (numeric) 19. euribor3m - Daily euribor 3-month rate - (numeric) 20. nr.employed - Quarterly number of employees - (numeric)   Output variable (desired target) – 21. Term Deposit - has the client subscribed a term deposit? - (binary: ‘yes’,‘no’)  Below I have included visuals of my data such as a bar chart, box plot, pie chart, etc. Visuals at the top are categorical. Visuals at the bottom are numerical. |
| **Preprocessing**: armed with the exploratory analysis, perform the necessary preprocessing, both general and specific types appropriate for the modeling type being employed. |
| Beginning with my data processing I had to determine what values need to either be replaced or dropped along with which values were missing. To start I had to determine my variable which mean I had to run the df.types code.  Text  Description automatically generated  Next, after assessing my variables and revisiting my categorical and numerical variable. I had to see what values were missing and by doing so I ran code df.isna().sum().  Text  Description automatically generated with medium confidence  I am not missing any values in my data but after following along in the video. It doesn’t hurt to get in the practice of running necessary codes that would have been ran in event I had to replace my missing variables. As a result of taking extra steps my data remained the same with no missing data.  Text  Description automatically generated |
| **Model Fitting**: explain the key steps and activities you perform to fit the model. Experiment (as appropriate) with parameters tuning. This is key, what separates highly accurate model from a less accurate ones is the amount of performance tuning performed. |
| Since, the dataset is clean I prepared are model fitting by running the sklearn.model\_selection import train\_test\_split X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size = 0.7, test\_size=.3). My train\_size was at 0.70% and my test\_size was 0.30% keep in mind that for the train and test size if float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size. I imported the GaussianNB() (an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring). Along with running nb.fit (Fit the model according to the given training data). Now to determine the coefficient of the prediction I am running nb.score(X\_test, y\_test). Getting an output of 0.685%. y\_pred = nb.predict(X\_test) is the last step in the modeling phase at this point I am predicting using the base regressor, applying inverse.  After implementing all these models’ accuracies will be compared using confusion matrix to determine best model for this dataset.  Text  Description automatically generated with low confidence |
| **Results** |
| **Model Properties:** explain the components of the fitted model and their characteristics. Leverage functions to summarize the model properties. Also, leverage visualization as required. |
| Dataset is first analyzed using bar plots bar charts to understand frequency distribution of the variables. Given below frequency bar plots of some attributes in the data set. Some attributes needed transformation to numeric class for fitting the models. Using cols = function that transformation was done. Then data is split in 7:3 ratio. 70% data is used for training the model and 30% for testing the model. After having training and testing dataset, we can now fit models. |
| **Output Interpretation**: explain the result and interpret the final model output using terms that reflect the application area and in relation to the stated objective. This is where you check whether or not the stated objective is met. |
| The stated objective has met. |
| **Evaluation**: employ appropriate metrics to quantitatively evaluate the performance of the fitted model. For supervised classification, this includes simple accuracy, precision & recall (or sensitivity & specificity), all of which can be generated from a confusion matrix, or ROC. |
| For Logistic Regression (LR):  Below is the results and confusion matrix for DT ✓ Accuracy: 0.73 ✓ Precision: [0:0.63, 1:0.83]- The number of true positives.  ✓recall: [0:0.90, 1:0.47]- The number of true positives. ✓ F1 Score: [0:0.74, 1:0.60]- The harmonic means of precision and recall  I have determined the accuracy is 69%.  Calendar  Description automatically generated  I did not have a ROC chart because I have a multiple classification dataset.  The Predicted value of 9902 is a true positive.  The predicted value of 5834 is a false negative.  The predicted value of 5122 is true negative.  The predicted value of 1071 is false positive. |
| **Conclusion** |
| **Summary**: highlight the main findings in relation to the stated objective. You don’t need to discuss the details of the analysis and the model such as accuracy here, just focus on the key findings. |
| After running multiple models on the dataset, Decision Tree is found to give best accuracy of 69% |
| **Limitations & Improvement areas**: discuss the limitations of the analysis and identify potential improvement areas for future work. This could be related to the data, algorithm, or a combination of the two. |
| If your test data set has a categorical variable of a category that wasn’t present in the training data set, the Naive Bayes model will assign it zero probability and won’t be able to make any predictions in this regard. This phenomenon is called ‘Zero Frequency,’ and you’ll have to use a smoothing technique to solve this problem. |

|  |
| --- |
| **Appendix** |
| <https://colab.research.google.com/drive/1BsS0Wg2m7uz_rxuilizJ6IFaV6EnWXWk?usp=sharing>  Reference Information & Descriptions  # This colab notebook provides an analysis of Bank Direct Marketing Campaigns  # This dataset can be acquired from this link: https://www.kaggle.com/datasets/ruthgn/bank-marketing-data-set?resource=download  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  Data Ingestion  df = pd.read\_csv('/content/bank-direct-marketing-campaigns.csv')  Set initial options  pd.set\_option('display.max\_columns', 40)  pd.set\_option('display.max\_rows', 40)  df  Initial Inspection & Data Cleansing  df.head()  df.tail()  df.shape  df.info()  df.dtypes  df.columns  df.isna().sum()  How many catergorical vs Numerical Columns  catergorical = df.select\_dtypes(include=[object])  print("catergorical Columns:",catergorical.shape[1])  numerical = df.select\_dtypes(exclude=[object])  print("Numerical Columns:",numerical.shape[1])  Data Exploration  df.describe()  df.describe(include='all')  df.describe(include='all', percentiles=[0.01, 0.5, 0.1, 0.9, 0.99, 0.995])  correlation  df.corr()  correlation heatmap  sns.heatmap(df.corr().abs(), annot = True,cmap = 'coolwarm')  Data Visualization  df.columns  df.dtypes  Catergorical Variable  df['job'].value\_counts()  df['job'].value\_counts().plot(kind='bar', ylabel="Frequency", title='gender')  df['job'].value\_counts().plot(kind='pie', ylabel="", title='job', autopct='%1.1f%%', startangle=90)  df['marital'].value\_counts()  df['marital'].value\_counts().plot(kind='bar', ylabel="Frequency", title='marital')  df['marital'].value\_counts().plot(kind='pie', ylabel="", title='marital', autopct='%1.1f%%', startangle=90)  df['education'].value\_counts()  df['education'].value\_counts().plot(kind='bar', ylabel="Frequency", title='education')  df['education'].value\_counts().plot(kind='pie', ylabel="", title='education', autopct='%1.1f%%', startangle=90)  df['default'].value\_counts()  df['default'].value\_counts().plot(kind='bar', ylabel="Frequency", title='default')  df['default'].value\_counts().plot(kind='pie', ylabel="", title='default', autopct='%1.1f%%', startangle=90)  df['housing'].value\_counts()  df['housing'].value\_counts().plot(kind='bar', ylabel="Frequency", title='housing')  df['housing'].value\_counts().plot(kind='pie', ylabel="", title='housing', autopct='%1.1f%%', startangle=90)  df['loan'].value\_counts()  df['loan'].value\_counts().plot(kind='bar', ylabel="Frequency", title='loan')  df['loan'].value\_counts().plot(kind='pie', ylabel="", title='loan', autopct='%1.1f%%', startangle=90)  df['month'].value\_counts()  df['month'].value\_counts().plot(kind='bar', ylabel="Frequency", title='month')  df['month'].value\_counts().plot(kind='pie', ylabel="", title='month', autopct='%1.1f%%', startangle=90)  df['day\_of\_week'].value\_counts()  df['day\_of\_week'].value\_counts().plot(kind='bar', ylabel="Frequency", title='day\_of\_week')  df['day\_of\_week'].value\_counts().plot(kind='pie', ylabel="", title='day\_of\_week', autopct='%1.1f%%', startangle=90)  df['poutcome'].value\_counts()  df['poutcome'].value\_counts().plot(kind='bar', ylabel="Frequency", title='poutcome')  df['poutcome'].value\_counts().plot(kind='pie', ylabel="", title='poutcome', autopct='%1.1f%%', startangle=90)  df['y'].value\_counts()  df['y'].value\_counts().plot(kind='bar', ylabel="Frequency", title='y')  df['y'].value\_counts().plot(kind='pie', ylabel="", title='y', autopct='%1.1f%%', startangle=90)  Numerical Variables  df['emp.var.rate'].plot(kind='hist')  df.boxplot(column='emp.var.rate', rot=90)  df['cons.price.idx'].plot(kind='hist')  df.boxplot(column='cons.price.idx', rot=90)  df['cons.conf.idx'].plot(kind='hist')  df.boxplot(column='cons.conf.idx', rot=90)  df['euribor3m'].plot(kind='hist')  df.boxplot(column='euribor3m', rot=90)  df['nr.employed'].plot(kind='hist')  df.boxplot(column='nr.employed', rot=90)  Data Preprocessing  df.dtypes  df.isna().sum()  # Use mode for catergorical variable  df.fillna(df.select\_dtypes(include='object').mode().iloc[0], inplace=True)  # Use mean for numberical values or median if there are strong outliers  df.fillna(df.select\_dtypes(include='number').mode().iloc[0], inplace=True)  df.isna().sum()  df  One Hot Encoding  df.columns  df.dtypes  # Don't encode the target column of 'y' (y)  cols = ['job', 'marital', 'education', 'default', 'housing', 'loan',  'contact', 'month', 'day\_of\_week', 'poutcome']  df = pd.get\_dummies(df, columns = cols, drop\_first=True)  df  Label Encoding  from sklearn.preprocessing import LabelEncoder  le = LabelEncoder()  df['y'] = le.fit\_transform(df['y'])  le.classes\_  df['y'].value\_counts()  df  Shuffle the Dataset  from sklearn.utils import shuffle  df = shuffle(df)  df  split into x and y  X = df.drop(['y'], axis=1)  X  y = df['y']  y  Balance the DataSet  from collections import Counter  from imblearn.over\_sampling import RandomOverSampler  # summerize class distribution  print(Counter(y))  # define oversampling strategy  oversample = RandomOverSampler(sampling\_strategy='minority')  # fit and apply the transform  X, y = oversample.fit\_resample(X, y)  # summerize class distribution  print(Counter(y))  Normalize the Dataset  from sklearn.preprocessing import MinMaxScaler  scaler\_m = MinMaxScaler()  X.shape  Train Test Split  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size = 0.7, test\_size=.3)  Modeling  from sklearn.naive\_bayes import GaussianNB  nb = GaussianNB()  nb.fit(X\_train, y\_train)  nb.score(X\_test, y\_test)  y\_pred = nb.predict(X\_test)  Evaluation  from sklearn.metrics import confusion\_matrix, classification\_report  cm = confusion\_matrix(y\_test, y\_pred)  cm  print(classification\_report(y\_test, y\_pred))  Plotting of Confusion Matrix  import seaborn as sns  plt.figure(figsize=(10,7))  sns.heatmap(cm, annot=True, fmt='g', cmap='Blues')  plt.xlabel('Predicted')  plt.ylabel('Truth') |

**References**