**Assignment 2**

Melissa Hunfalvay

Email: [Melissa.Hunfalvay@gmail.com](mailto:Melissa.Hunfalvay@gmail.com)

Data 650 9040

Summer 2022

Professor Ozan Ozcan

Date: June 28th, 2022

**Part 1**

**Question 1**

*Discuss the importance of data exploration and visualization prior to running the logistic regression method.*

A key component of the data analytics lifecycle includes data exploration (EMC Educational Services, 2015; Erl, Khattak, Buhler, 2016). A data scientist is responsible for conducting data exploration. Understanding the data, prior to analysis allows the analyst to:

1. Understand basic characteristics of the data, such as the number of passengers (rows) within the data set for the titanic datafile.
2. Discover the variables which may be used to formulate an analytical question, such as the column of “Survived” persons which is classified as 0,1. This indicate an ability to explore predictive questioning.
3. Create a baseline understanding of the data from which to continually refer throughout the analytical process. Specifically, the number of passengers in the survived versus did not survive is unequally distributed. This may affect decisions in the data cleaning process especially when the data is subcategorized by passenger class. Unequal distribution may impact future cut off values in the analytic process.

A technique for better data exploration, especially of large datasets, is to use visualizations. Visualizations take many forms such as maps or graphs. However, what they have in common is a way to see the data in a manner outside of a datafile by providing a visual context.

Visualizations are used to:

1. Make it easier to see trends in the data such as the multicolored bar graph in the titanic data which shows survival rates by socio-economic class (Figure 1). In this visualization, we can clearly see that passengers in 3rd class were almost three times more likely to not survive compared to 1st class passengers who were more likely to survive. This could be important to keep in mind later in the analytical process when looking at regression weighted contributors to the model. This insight into the data would not have been known without a visual synthesis of results.

Chart, bar chart

Description automatically generated

Figure 1: Survival of Titanic passengers by economic class

1. Determine skewness of variables using distribution graphs such as the age distribution graph showing a mostly right skewed distribution (Figure 2). This is important as skewed variables may affect different analytical techniques and may need to be ‘corrected’ or dropped in the data cleaning phase.

Chart, histogram

Description automatically generated

Figure 2: Age distribution of all passengers who traveled on the Titanic

In summary, data exploration and the use of visualizations as a tool to glean insights into data are critical components for analysts to understand the data and make appropriate decisions throughout the data analytics lifecycle.

**Question 2**

*What do the Titanic data exploration results reveal about the relationships between the likelihood of survival and passenger data?*

Three main insights can be gleaned from the data exploration results about the relationship between the likelihood of survival and passenger data:

1. More passengers died in the titanic sinking (n = 443) than survived (n = 313, Figure 3).

A picture containing text, antenna, screenshot

Description automatically generated

Figure 3: Table showing the survival count for passengers on the Titanic

1. The class a passenger was ‘may be’ a predictor of survival in the regression model results conducted later in the analysis. Specifically, first class passengers have more people surviving than perishing. 3rd class passengers have almost three times more likelihood of perishing than surviving (Figure 1 and 4).

Table

Description automatically generated

Figure 4: Table of the frequency of survived versus perished passengers on the Titanic

1. More passengers were younger (right skewed) than older on the titanic (Figure 5). The highest number of persons where aged in the early to late twenties (Figure 5). These were also the highest number of passengers who died (Figure 6).

Chart, histogram

Description automatically generated

Figure 5: distribution of age across all passengers on the titanic.

Highest number of passengers who died was between mid to late twenties and corresponds to the highest number of passengers on the Titanic

Chart, histogram

Description automatically generated

Figure 6: Distribution of age for passengers who died on the titanic

Additional interesting insights include:

1. Approximately 45 children aged below about 8 years of age were on the ship (Figure 6). Most of those passengers survived (approximately 35, Figure 7) while a disproportionately small number (n = 10) died. This trend is not the case for people who are aged, as persons 65 or older almost all perished in the sinking (Figure 6 & 7).

Chart, histogram

Description automatically generated

Of the approximately 13 older individuals (65+ years) all died

Fewer young children died on the Titanic, approximated to be 22%

Figure 6: Age distribution of passengers on the Titanic

Chart, histogram

Description automatically generated

Figure 7: Ade distribution of passengers who died on the Titanic

1. The titanic had the fewest passengers in first class, then 2nd class and the most passengers in third class (Figure 8).

Table

Description automatically generated

Figure 8: Passengers who survived (0) and died (1) on the Titanic by passenger class

**Question 3**

*Discuss the logistic regression method results, including the classification accuracy for training and test set.*

After exploration of the data the methodology for logistic regression begins.

*Step 1: Parse Data and Create Data Frame for Spark RDD*

* Identifying the total number of records (n = 757)
* Adding the inferSchema = true ensures that the columns in the data frame have the correct data types (Figure 9).
* Printing the first 5 rows for inspection (Figure 9). Note the data is originally in a csv format. It is read in as an RDD (Figure 9). SC refers to Spark Context. An RDD is a “sparse, distributed, persistent multidimensional sorted map” that can be read by a distributed database (Khurana, 2012).

Text

Description automatically generated

Figure 9: Shows the variable names (row 1) and the first 4 rows of passenger data

* Parse the data using the def parseDocument function. This changed the RDD into a Spark data frame.
* Text value refers to predictor variables numbers PClass (variable 1), age (variable 2), and sex (variable 3).
* The function returns a row with the passenger unique ID, the group of predictor variables (textValue) and the target variable (Survived).

Text

Description automatically generated

Figure 10: Function that parses data into dataframe

* Add the parsed data to a dataframe: TitanicData **=** documents**.**toDF()
* Return the number of record (n = 756)
* Print the first 5 rows of data to ensure the datafile looks accurate (Figure 11). Note this can be compared to Figure 9 for a visual conformation of accuracy of the cells. The datafile is formatted differently because the file is now formatted for Spark distributed database.

Text

Description automatically generated

Figure 11: Shows the data after it has been parsed and is now formatted as a “big data” distributed file.

Note that in figure 10 the first row included the column names whereas in figure 11, which is the data after it has been parsed and added to the data frame it no longer has column headers. This is also why the total number of records prior to parsing the data was one more 757 compared to the 756 after parsing.

The Spark data frame is configured as a key-value store. The key is the prior csv file column, and the value, is the rest of the data in that column (Figure 12). Each row is identified uniquely by a row key.

Values

Keys

Text

Description automatically generated

Unique Identifiers

Figure 12: Shows the unique identifier, keys and values for the new Spark distributed big data file.

*Step 2: Divide Data for Training and Testing*

* Data is divided into 80% for training (n = 597) and 20% for testing (n = 159).
* A check of the first 20 rows of the training data set is then printed for review.
* The purpose of the training data is for the algorithm within the model to test itself against a dataset to determine its structure or “goodness of fit.”
* The purpose of the test data is to see if the results from the training data can generalize to a new dataset, i.e., do the results hold up when testing against unseen data.

*Step 3: Build the Model*

* To build the Logistic Regression model we use the Spark Machine learning library (<https://spark.apache.org/mllib/>, Figure 13).

Graphical user interface, text, application, email

Description automatically generated

Figure 13: Packages imported from Spark Machine learning library

* The Pipeline for model development consists of three stages (figure 14):
  1. Tokenizer: splits PersonInfo into words and adds words column to data frame
  2. HashingTF: converts the new words column into feature vectors.
  3. Method call: the pipeline calls the LogisticRegression.fit() method to produce a LogisticRegressionModel

*Text

Description automatically generated*

Figure 14: Explains the stages of model development

* Figure 15 shows the order in which the pipeline is to be run on the training data.

Logo

Description automatically generated with medium confidence

Figure 15: Pipeline order for training data

*Step 4: Predict for Test Data*

* Output The output for five of the 159 test records are shown in Figure 17. Information from the columns of PersonInfo is shown first, followed by the prediction result for that individual (1 = survived, 0 = died) and then the probability of the result. For example, in row one the result was the person died (prediction = 0), the probability of died was 96% (0.9552) and the probability of survived was 4% (0.0448) according to the logistic regression model. The default threshold for probability is .5.

A picture containing text

Description automatically generated

A picture containing text

Description automatically generated

Figure 17: Model predictions for the first 5 rows of data

* Results are then tabulated for the models predicted outcome (Figure 18) versus the actual outcome (Figure 19).

Table

Description automatically generated

Figure 18: Shows the models predicted outcome for the test data

Table

Description automatically generated with medium confidence

Figure 19 shows the actual outcome for the test data

* A classification table is then created (Figure 20) which shows the number of correctly and incorrectly classified passengers based on the logistic regression model.

Table

Description automatically generated with low confidence

Figure 20: Correct and incorrect classification of test data for Titanic data set. Green boxes show correctly classified data, whereas red boxes show incorrectly classified data

*Step 5: Evaluate the Model*

1. *Evaluate the Model on the Training dataset*

Text

Description automatically generated

Figure 21: Model evaluation statistics for the training data

Calculations are then performed on the training dataset to determine the overall accuracy, inaccuracy (error), precision, recall and F1 scores (Figure 21).

*Accuracy* is calculated by the number of correctly classified instances divided by the total number of instances/rows. The goal of the model is to have as many correctly classified instances as possible. For the training data the accuracy was high at 82% (Figure 21).

*Error* refers to the incorrectly classified instances. The goal of the model is to have as few correctly classified instances as possible and is the inverse of accuracy. For the training data the error was 18%. For the Titanic dataset this is a good error rate (Figure 21).

*Precision* is the number of true positives over the number of predicted positives. In other words, the models predicted positive results. Formula: true positive/(true positive + false positive). Precision for Titanic data is good at 81% (Figure 21).

*Recall/Sensitivity Rate /True Positives:* The ability of the model to identify those who survived. Formula: true positive/(true positive + false negative). The training dataset sensitivity was 77%. This is good and certainly better than without the model (Figure 21).

*F1 Score:* combines precision and recall F1 score is best when close to 1 and worst when zero. Formula: 2 \* ((Precision \* recall)/precision + recall)). The result for the training data was 0.79 which is a good result.

1. *Evaluate the Model on the Test Dataset*

The same calculations that were conducted on the training set are now performed on the test set of data (Figure 21). Results show a 4% difference in accuracy between the training and test datasets (training = 82%; test = 78%) and subsequently in the error (training = 18%; test = 22%). Change in precision is 14% reduction! Note this also relates to the area under the PR (next section).

There is also a slight reduction in the recall (2.5%). F1 measure remains above .7 which is good. In summary the model generalized fairly well based on accuracy scores, however, the negative is that the ability of the model to generalize its predictions of positive cases (i.e., survival) was significantly reduced across new data.

Text

Description automatically generated

Figure 21: Model evaluation statistics for the test data

1. *Evaluate the Model using the ROC and PR statistics*

*The area under Precision Recall:* (PR, Figure 22) is particularly useful when classes are imbalanced. In the case of the Titanic data there was 313 (41%) people who survived and 443 (59%) who died. The data is moderately skewed toward those who died.

The reason for using area under the PR is that typically the large number of class 0 examples means we are less interested in the skill of the model at predicting class 0 (died) correctly, e.g. high true negatives. The area under the PR curve is especially useful in machine learning for evaluating binary classification methods such as the titanic dataset.

Key to the calculation of precision and recall is that the calculations do not make use of the true negatives. It is only concerned with the correct prediction of the survival, class 1. This metric can be helpful when reviewing several different machine learning models where determining which model to for binary classification with imbalanced data is most useful.

The result of .62 for area under PR is poor (Hastie, Tibshirani, Friedman, 2017). Therefore, the results should be cautioned when trying to predict those who survived.

*Area Under the Curve (AUC):* The AUC (Figure 22) is used to summarize the performance of each classifier (1 = survived, 0 - died) into a single measure. The AUC is a measure that is equivalent to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance ([Chan,](https://www.displayr.com/what-is-a-roc-curve-how-to-interpret-it/) 2020). It is equivalent to a Wilcoxon rank sum statistic. The high AUC score indicates better classification. AUC score for this dataset was 0.77 (Figure 22) which is fair (Hastie, Tibshirani, Friedman, 2017).



Figure 22: Area under PR and ROC statistics

1. *Evaluate the model by plotting the ROC curve*

The first 20 ROC curve points are printed (Figure 23) and the visual ROC curve plot is printed (Figure 24). *Receiver Operating Curve (ROC):* is a visualization that displays the false positive rate (specificity; X-axis) and the true positive rate, (sensitivity; Y-axis). The ROC shows the trade-off that occurs between these two measures. The closer the curve to the top left corner the higher the probability that the model correctly predicts the target variable. When the curve is close to the 45-degree (red) diagonal line the probability that the model correctly predicts the target variable is low (chance prediction).

*Table

Description automatically generated*

Figure 23: False positive and true positive rates for a portion of the first 20 coordinates on the ROC curve

Chart, line chart

Description automatically generated

Figure 24: ROC curve visualization with a .5 chance line added for context

**Question 4**

*Is logistic regression suitable for this problem? Why or Why not?*

Logistic regression is a form of classification that can be used to predict an outcome. In the case of the titanic dataset, it can be used to ask: “Can we predict who will survive the titanic sinking?” It is often used for target variables with binary classification outcomes.

Technically, it is a model than can be used to evaluate this data. However, the better question is probably should it be the model that is put in a production environment. The answer is no. Why? Because it does not do a good enough job of predicting those who survived.

**Question 5**

*What alternative machine learning methods could be suitable for this problem? Consider at least 2 alternative methods.*

There are three supervised learning classification models I would consider as alternatives to the logistic regression. These are:

1. Artificial Neural Network (ANN): because the ANN can implement tasks that are not linear in nature and therefore provide an alternate view from which to evaluate a model.
2. Support Vector Machines (SVM): because, like ANN this formula take a non-linear alternate approach to classifying the data. SVM tries to find the best margin or distance between the line and support vectors therefore attempting to reduce error. As most of the error is coming from false classification of survivals this model is worth exploring.
3. Random Forest (RF): because the RF can have an impact on the true positive rate and therefore precision. I would choose this option last however, as the number of variables in this model are low in terms of the explanatory variables and RF benefits are unlikely with few explanatory variables.

**Part 2**

**Question 1**

*Define the purpose of the study and the target variable. Which variables are used as predictors?*

The purpose of the study is to predict birth weight in newborn children based on information about the mother.

The target variable is called “Low Birth Weight” and is a binary variable classify birthweight as either 0: >= 2500 grams, or 1: < 2500 grams.

The predictor variables are:

1. Mothers age in years (Age)
2. Race (1: White, 2: Black, 3: other)
3. Smoking status of the mother during pregnancy. Smoke; 1: no, 2: Yes
4. Mothers’ history of premature labor, PTL; 1: None, 2: One, 3: Two, etc.
5. History of hypertension HT; 1” No, 2: Yes.
6. Presence of Uterine irritability. UI: 1: No, 2: Yes
7. Number of physician visits during the first trimester. FTV; 1: None, 2: One, 3: Two, etc

**Question 2**

*Interpret the data exploration and visualization results. What did you learn about the low birth weight data from data exploration, including possible relationships between predictors and the target variable?*

Figure 1 shows descriptive statistics for all variables. Count shows no data is missing. The average age of the mothers was 23.24 (*SD* = 5.30). Most mothers did not experience hypertension (HT: mean: 0.06). Most mother did have at least one visit with their physician during the first trimester of the pregnancy (FTV: *mean* = 0.79, *SD* = 1.06).

Graphical user interface, application, table

Description automatically generated

Figure 1: Descriptive statistics for all variables in the low Birth Weight dataset

The number of rows in the data was 189, of which 59 of the children were < 2500 grams and 130 were >= 2500 grams (Figure 2 and Figure 3). Therefore, the target variable is skewed, whereby the low weight births are rarer cases.

Table

Description automatically generated

Figure 2: Frequency of target variable 0: >= 2500 grams, or 1: < 2500 grams

Calendar

Description automatically generated with medium confidence

Figure 3: Distribution count of target variable 0: >= 2500 grams, or 1: < 2500 grams.

Distribution plots also show the age of the mother is right skewed (Figure 4).

Chart, histogram

Description automatically generated

Figure 4: Distribution plot of mother’s age

There are very few black mothers in the dataset as seen in figure 5. This is noted in terms of a limitation of the analysis due the dataset.

Chart

Description automatically generated

Figure 5: Race frequency: 1: White, 2: black, 3: other.

Very few mothers had more than 2 visits with the doctor in the first trimester of the pregnancy (Figure 6). A high number of women (n = 100, note red box on Figure 6) did not visit the doctor at all during the first trimester.

A picture containing text, crossword puzzle

Description automatically generated

Figure 6: Number of physician visits during first trimester.

Only one woman visited the doctor 6 times in the first trimester (Figure 7).

Chart, scatter chart

Description automatically generated

Figure 7: Scatterplot showing each women’s number of physician visits in first trimester.

Sixty percent of mothers who had children with low birth weight did not see the doctor in the first trimester (Figure 8). This should be flagged as a potential predictor of significance.

Chart

Description automatically generated

Figure 8: Number of physician visits for mother with low birth weight.

Past research on low birth weight shows women who smoke tend to have lower birth weight babies (Centers for Disease Control, 2022). Figure 9 shows that of the women who had low birth weight babies in this dataset about half were smokers. This should be noted as a possible contributing predictor variable to model results.

Calendar

Description automatically generated

Figure 9: Birthweight of babies where the mother was a smoker.

Correlation matrix reveals no variables are highly correlated (Figure 10).

A picture containing table

Description automatically generated

Figure 10: Correlation matrix from -1.0 to +1.0

**Question 3**

*Discuss the method results, including the classification accuracy for training set and test set and model evaluation metrics (precision, recall, ROC curve area).*

After exploration of the data the methodology for logistic regression begins.

*Step 1: Parse Data and Create Data Frame for Spark RDD*

* Identifying the total number of records (n = 7190)
* Adding the inferSchema = true ensures that the columns in the data frame have the correct data types (Figure 9).
* Printing the first 5 rows for inspection (Figure 9). Note the data is originally in a csv format. It is read in as an RDD (Figure 9). SC refers to Spark Context. An RDD is a “sparse, distributed, persistent multidimensional sorted map” that can be read by a distributed database (Khurana, 2012).

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Figure 9: Shows the variable names (row 1) and the first 4 rows of mothers data

* Parse the data using the def parseDocument function. This changed the RDD into a Spark data frame.
* Text value refers to predictor variables numbers 2-8 (Figure 10)
* The function returns a row with the unique ID, the group of predictor variables (textValue) and the target variable (Low).

Text

Description automatically generated with medium confidenceA picture containing text

Description automatically generated

Figure 10: Function that parses data into dataframe

* Add the parsed data to a dataframe: lowbwt **=** documents**.**toDF()
* Return the number of record (n = 189)
* Print the first 5 rows of data to ensure the datafile looks accurate (Figure 11). Note this can be compared to Figure 9 for a visual conformation of accuracy of the cells. The datafile is formatted differently because the file is now formatted for Spark distributed database.

Text

Description automatically generated

Figure 11: Shows the data after it has been parsed and is now formatted as a “big data” distributed file.

Note that in figure 10 the first row included the column names whereas in figure 11, which is the data after it has been parsed and added to the data frame it no longer has column headers. This is also why the total number of records prior to parsing the data was one more 190 compared to the 189 after parsing.

The Spark data frame is configured as a key-value store. The key is the prior csv file column, and the value, is the rest of the data in that column (Figure 12). Each row is identified uniquely by a row key.

Keys

Values

Text

Description automatically generated

Unique Identifiers

Figure 12: Shows the unique identifier, keys and values for the new Spark distributed big data file.

*Step 2: Divide Data for Training and Testing*

* Data is divided into 80% for training (n = 148) and 20% for testing (n = 41).
* A check of the first 20 rows of the training data set is then printed for review.
* The purpose of the training data is for the algorithm within the model to test itself against a dataset to determine its structure or “goodness of fit.”
* The purpose of the test data is to see if the results from the training data can generalize to a new dataset, i.e., do the results hold up when testing against unseen data.

*Step 3: Build the Model*

* To build the Logistic Regression model we use the Spark Machine learning library (<https://spark.apache.org/mllib/>, Figure 13).

Graphical user interface, text, application, email

Description automatically generated

Figure 13: Packages imported from Spark Machine learning library

* The Pipeline for model development consists of three stages (figure 14):

1. Tokenizer: splits PersonInfo into words and adds words column to data frame
2. HashingTF: converts the new words column into feature vectors.
3. Method call: the pipeline calls the LogisticRegression.fit() method to produce a LogisticRegressionModel

*Text

Description automatically generated*

Figure 14: Explains the stages of model development

* Figure 15 shows the order in which the pipeline is to be run on the training data.

Logo

Description automatically generated with medium confidence

Figure 15: Pipeline order for training data

*Step 4: Predict for Test Data*

* The output for five of the 41 test records are shown in Figure 17. Information from the columns of PersonInfo is shown first, followed by the prediction result for that individual (0: >= 2500, 1: < 2500 g) and then the probability of the result. For example, in row one the result was the child was born with a high birth weight (prediction = 0), the probability of 0: >= 2500 birth weight was 88% (0.88) and the probability of a birth weight < 2500 g was 12% (0.12) according to the logistic regression model. The default threshold for probability is .5.

A picture containing text

Description automatically generated

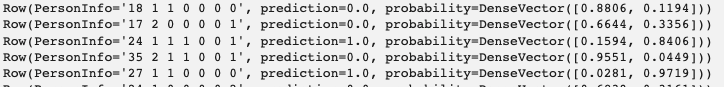


Figure 17: Model predictions for the first 5 rows of data

* Results are then tabulated for the models predicted outcome (Figure 18) versus the actual outcome (Figure 19).

Table

Description automatically generated

Figure 18: Shows the models predicted outcome for the test data

A picture containing text, antenna

Description automatically generated

Figure 19 shows the actual outcome for the test data

* A classification table is then created (Figure 20) which shows the number of correctly and incorrectly classified birthweights based on the logistic regression model.

Table

Description automatically generated with medium confidence

Figure 20: Correct and incorrect classification of test data for birthweight data set. Green boxes show correctly classified data, whereas red boxes show incorrectly classified data

*Step 5: Evaluate the Model*

1. *Evaluate the Model on the Training dataset*

Text

Description automatically generated

Figure 21: Model evaluation statistics for the training data

Calculations are then performed on the training dataset to determine the overall accuracy, inaccuracy (error), precision, recall and F1 scores (Figure 21).

*Accuracy* is calculated by the number of correctly classified instances divided by the total number of instances/rows. The goal of the model is to have as many correctly classified instances as possible. For the training data the accuracy was high at 81% (Figure 21).

*Error* refers to the incorrectly classified instances. The goal of the model is to have as few correctly classified instances as possible and is the inverse of accuracy. For the training data the error was 19%. For the birthweight dataset this is a good error rate, however, given the nature of the data I would want to reduce this error and dig deeper into the data (Figure 21).

*Precision* is the number of true positives over the number of predicted positives. In other words, the models predicted positive results. Formula: true positive/(true positive + false positive). Precision for birthweight data is good at 77% (Figure 21).

*Recall/Sensitivity Rate /True Positives:* The ability of the model to identify those who with low birthweight. Formula: true positive/(true positive + false negative). The training dataset sensitivity was 52%. This is poor and according to statistical standard would fail as it only detects low birth rate at about a rate of chance, a coin toss (Hastie, Tibshirani, Friedman, 2017, Figure 21).

*F1 Score:* combines precision and recall F1 score is best when close to 1 and worst when zero. Formula: 2 \* ((Precision \* recall)/precision + recall)). The result for the training data was 0.62 which is a fair to moderate result.

1. *Evaluate the Model on the Test Dataset*

The same calculations that were conducted on the training set are now performed on the test set of data (Figure 21). Results show a 27% difference in accuracy between the training and test datasets (training = 81%; test = 54%) and subsequently in the error (training = 19%; test = 46%). This model is significantly overfitting from training to test data. It is also not sensitive enough to predict low birth weights, which are the rarer instances in which we hope to detect. This model does not pass to production and alternative needs to be considered.

Change in precision is 43% reduction! Note this also relates to the area under the PR (next section).

Recall is poor in both training and test datasets (<53%). F1 translates poorly in the test data (0.30).

In summary, and as stated previously, this model fails to correctly detect low birth weight in these mothers. This is a classic case of overfitting training data which showed a good accuracy 81% but very poor generalizability in test data (53%).

Text

Description automatically generated

Figure 21: Model evaluation statistics for the test data

1. *Evaluate the Model using the ROC and PR statistics*

*The area under Precision Recall:* (PR, Figure 22) is particularly useful when classes are imbalanced. In the case of the low birthweight data there was 59 (31%) children with low birthweight and 130 (69%). The data is heavily skewed toward those who with low birthweight.

The reason for using area under the PR is that typically the large number of class 0 examples means we are less interested in the skill of the model at predicting class 0 (>=2500g) correctly, e.g. high true negatives. The area under the PR curve is especially useful in machine learning for evaluating binary classification methods such as the low birthweight dataset.

Key to the calculation of precision and recall is that the calculations do not make use of the true negatives. It is only concerned with the correct prediction of the low birthweight, class 1. This metric can be helpful when reviewing several different machine learning models where determining which model to for binary classification with imbalanced data is most useful.

The result of .34 for area under PR is a fail (Hastie, Tibshirani, Friedman, 2017). Therefore, the results should not be used when trying to predict low birthweight.

*Area Under the Curve (AUC):* The AUC (Figure 22) is used to summarize the performance of each classifier (1: <2500g, 0: >=2500g) into a single measure. The AUC is a measure that is equivalent to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance ([Chan,](https://www.displayr.com/what-is-a-roc-curve-how-to-interpret-it/) 2020). It is equivalent to a Wilcoxon rank sum statistic. The high AUC score indicates better classification. AUC score for this dataset was 0.48 (Figure 22) which is a fail (Hastie, Tibshirani, Friedman, 2017).



Figure 22: Area under PR and ROC statistics

1. *Evaluate the model by plotting the ROC curve*

The first 20 ROC curve points are printed (Figure 23) and the visual ROC curve plot is printed (Figure 24). *Receiver Operating Curve (ROC):* is a visualization that displays the false positive rate (specificity; X-axis) and the true positive rate, (sensitivity; Y-axis). The ROC shows the trade-off that occurs between these two measures. The closer the curve to the top left corner the higher the probability that the model correctly predicts the target variable. When the curve is close to the 45-degree (red) diagonal line the probability that the model correctly predicts the target variable is low (chance prediction).

*Table

Description automatically generated*

Figure 23: False positive and true positive rates for a portion of the first 20 coordinates on the ROC curve

Chart, line chart

Description automatically generated

Figure 24: ROC curve visualization with a .5 chance line added for context

**Question 4**

*Is the logistic regression method suitable for this study? Why or why not?*

Logistic regression is a form of classification that can be used to predict an outcome. In the case of the low birthweight dataset, it can be used to ask: “Can we predict which mother will have children who have low birthweight?” It is often used for target variables with binary classification outcomes.

Technically, it is a model than can be used to evaluate this data. However, the better question is probably should it be the model that is put in a production environment. The answer is no. Why? Because the model is overfitting data and will not generalize well.

**Question 5**

*How would you improve the accuracy of your model?*

Given more time for this project I would experiment with the following options for the logistic regression model:

1. Hyperparameter tuning of predictor variables: one example could be to reduce variables and using smokers, age, and number of visits to physician
2. I would recommend adding more data if possible
3. Run ANOVA’s on the predictor variables to see sig differences are found in order to drive which variable to consider for the model
4. Consider changing the percentage of training and test data. I had a short time to experiment with this option, but overfitting still occurred. Therefore, I would do this after modifying predictor variables.
5. Add Ensemble and Boosting algorithms (also discussed next)

**Question 6**

*Discuss at least 2 alternative machine learning methods that could be suitable for this problem and explain why?*

There are five supervised learning classification models I would consider as alternatives to the logistic regression. These are:

1. Ensemble models: which are models of models and can achieve higher accuracy and robustness and better generalization than a single model and is particularly effective in smaller sample sizes such as the low birthweight data.
2. Boosting algorithms: such as AdaBoost (Adaptive Boosting), Gradient tree boosting and XGBoost. Boosting algorithms increase the accuracy of models. The algorithm creates random samples of the data and calculates prediction for each sub-sample. The model then learns from these subsamples and turns weak predictors into strong learners.
3. Random Forest (RF): because the RF can have an impact on the true positive rate and therefore precision. RF benefits can occur when there are higher number of explanatory variables.
4. Artificial Neural Network (ANN): because the ANN can implement tasks that are not linear in nature and therefore provide an alternate view from which to evaluate a model.
5. Support Vector Machines (SVM): because, like ANN this formula take a non-linear alternate approach to classifying the data. SVM tries to find the best margin or distance between the line and support vectors therefore attempting to reduce error. As most of the error is coming from false classification of survivals this model is worth exploring.

Note: I would also consider hyperparameter tuning within these models to improve performance.

**Part 3**

**Question 1**

*What is overfitting? What is the impact of overfitting on model performance? Discuss at least 2 approaches to avoid overfitting the model.*

Before defining overfitting, it is important to understand the concept of “goodness of fit” which refers to how well the data model matches the actual results of the data. This is best shown graphically/visually. Figure 1 shows a well fit model because the green dots (actual data) and the purple line show a similar pattern.

Chart, scatter chart

Description automatically generated

Figure 1: Well fit machine learning model

The principle of parsimony (also known as Occam’s razor) states that “entities [data] should not be multiplied beyond necessity” (Schaffer, 2015). Overfitting is the use of models that violate this principle. Figure 2 is an example of overfitting a model to a specific dataset.

Chart

Description automatically generated

Figure 2: An overfit model of data

One may think this model is a better representation of the data. It is. BUT, for this dataset only. In other words, it is too dependent on this dataset. Hence, the true problem of overfitting comes when the model is used on data that it was not trained or built on. The test dataset is split from the original data for the purpose of “testing” the generalizability of the model. As data scientists we want our models to be used on new data, and we want them to work well on the new data, this concept referred to as “generalizability”. An overfit model does not reflect the population of data.

To avoid overfitting the bias-variance trade-off should be considered. The bias-variance tradeoff is the property of a model that the variance of the parameters across samples can be reduced by increasing bias (Kohavi & Wolpert, 1996). Bias refers to an error from an erroneous assumption in the model. Variance refers to error from sensitivity to small fluctuations in the training set. A high variance can result in modeling “noise” in the data and leading to overfitting. The goal is to choose a model with both accuracy in the training data that also generalizes to new data.

One solution to overfitting is dimension reduction. Dimension reduction is a process of reducing variance in the data while still keeping a low-dimension representation of that data to retain meaningful properties.

A second solution is feature selection. Feature selection refers to selecting a subset of input variables for model construction. One method for choosing variables for a model that I have used in the past is to first run ANOVA’s on the input variables and remove any from further processing if they are not significant.

Other options may include adjusting the volume of data. Often times, small data sets have difficulty generalizing and with more data for the model to train on they can be generalized. On the flip side, very large and noisy dataset should be reduced as there may be features that show significance erroneously and impact the model’s effectiveness.

Finally, as the bias-variance trade-off are at odds with one another it is often the responsibility of the data scientist to deeply understand the model, the data and the question being asked to make correct decisions when tuning model parameters. This can be a preventative measure, for overfitting the model.

**Question 2**

*Discuss 5 (five) key differences between HDFS and Object Storage.*

Big data requires “big storage.” HDFS is the Hadoop Distributed File Storage for large data. Object storage is another method of storing data. One example is AWS S3 which stores data as objects with keys, values, and versions. Table 1 outlines the key differences between these two storage methods. A rigorous debate in the technology community has occurred as to the pros and cons of these different methods.

This is a well-known (and debated) issue in my company. We store data in AWS S3 buckets; however, I cannot directly access or change the data in that bucket. Instead, to access data I need to write a Python script to pull data from a combination of S3 and our Relational Database System (RDS) The advantage for RightEye of using S3 buckets for storage is secure and reliable with good backup systems should an AWS region go down. The disadvantage is a lack of transparency and direct access to the data for data science and analytical purposes.

Table 1: Key Differences between HDFS and Object Storage

|  |  |  |
| --- | --- | --- |
| **#** | **HDFS** | **Object Storage** |
| 1: Storage | Operates on commodity hardware. This means big data storage at a relatively cheaper price. | Operates on high-end storage servers. This means storage can be expensive as it is on custom machines. |
| 2: Scalability | Is horizontally flexible for scalability where clusters run on few nodes to many with easy “spin-up” when needed. | Is more vertically flexible for scalability since it is not easy to add servers. |
| 3: Architecture | HDFS has both storage and compute components. Name nodes and data nodes are connected allowing for queries that can access the data through the architecture which in turn allows for computing on the data. | Object storage, such as AWS S3 buckets, are only suited to storage. There is no ability to compute. This is problematic for an analyst trying to understand the data but can be advantageous to the platform architecture as the decoupled storage provides architectural flexibility. |
| 4: Data Types | HDFS works with structured, unstructured, and semi-structured data. | Object storage works well with structured and semi-structured data (Triniti, 2022). |
| 5: Amending Files | HDFS can append data files, that is add records to the end of a table. | Object storage works well with the entire file through creating and deleting the file. It does not append within a file. |
| 6: Storage | HDFS replicates data to create redundancies in case of failures. Failures are more frequent due to the data being stored on commodity hardware. This further increases the storage “footprint” of the data. | Failures are less prevelent in Object Storage because of server quality. Object storage uses Erase Coding which breaks data into fragments to store at different locations. This reduces the need for data replication. Furthermore, in AWS create data availability in different time zones which reduces the latency of user interaction across the world (RightEye has experienced this phenomenon). |
| 1. Cost | HDFS has higher utility costs than Object storage as HDFS requires more processors, memory, and storage to provide horizontal scalability. This requires more on-site electricity and cooling systems. | Object storage requires less hardware but more expensive hardware. Ongoing utility costs are lower however due to less electricity and physical storage needs. |

**Question 3**

*We may use R, Python, Scala, and Java programming languages with Spark. Discuss the pros and cons of each language.*

The pros and cons of R programming language are documented in table 1. In summary, there are many more positives to R than negatives in my opinion. As a statistician I appreciate the deep capabilities of R libraries. Furthermore, as I am a beginner programmer, the fact that R is different than other languages make little difference to be as I need to learn each language.

Table 1: Pros and Cons of R Programming Language

|  |  |  |
| --- | --- | --- |
| **#** | **Pros of R** | **Cons of R** |
| 1 | Created by statisticians and therefore has excellent Statistical Computing and Analysis | R syntax is very different than other languages this makes the learning curve long. |
| 2 | Open Source so anyone can use it | As the R syntax is very different than other languages it makes the transferability of learning this language less generalizable to other languages |
| 3 | Has a large variety of libraries and applications | R datatypes are very different than other languages |
| 4 | R is machine independent i.e. it supports cross-platform operation to include windows, mac, Linux, Java, Hadoop | Some developers state that R can make some easy things hard such as unique syntax requirements |
| 5 | Supports various data types such as vectors, arrays, matrices | Although R has many packages, some may be redundant, others may be of poor quality. Therefore, programmers need to know what to use. |
| 6 | Can do various stages of the Data Analytics lifecycle such as cleaning, wrangling and web scraping | R can run slow because of poor memory management which can take up computer space. |
| 7 | Can produce production level graphics and visualizations that can be static or dynamic | R lacks basic security measures. This may be important in some instances such as web-apps which may not be safe. |
| 8 | Has a highly active community which means high usage and bugs, or issues are brought forward quickly for resolution | R has no dedicated support team. Support is done via a community approach which may provide many answers, and some may not be useful to solving the problem. It will often fall on the developer to do trial-and-error to solve a problem which can be inefficient. |
| 9 | R can be used in parallel and distributed computing using libraries such as ddR and multiDplyr (TechVidan, 2022). | Although R is a flexible language this also means it can lack structure as there are no strict guidelines to follow. |
| 10 | Does not need a compiler because it is an interpreted language meaning it does not need a compiler to change code into an executable program |  |
| 11 | R is compatible with other languages such as C and C++ |  |
| 12 | Given that it is created by statisticians it can also be sued for machine learning applications such as sentiment analysis |  |
| 13 | R can interact with databases via packages such as Roracle |  |
| 14 | R has a very comprehensive development environment for both statistical computing and software development |  |
| 15 | R can produce fully fledged web applications using Rshiny |  |
| 16 | R is an object-oriented programming language which is an advantage because it allows the programmer to break the program into bite sized chunks (one object at a time) improving productivity and quality |  |
| 17 | Although R can make some easy things hard, the pro is that add-on packages can make easy things easy as well, such as SAS or SPSS. |  |

The pros and cons of Python programming language are documented in table 2. In summary, Python, like R has some similar pros and cons. Python has many more advantages than disadvantages and is a very popular language used by 41.6% of programmers (Berkeley Extension, 2022). Writing code using Python includes many developers that continues to grow which makes it continually viable for future use.

Table 2: Pros and Cons of Python Programming Language

|  |  |  |
| --- | --- | --- |
| **#** | **Pros** | **Cons** |
| 1 | Python is easy to learn (Szkaradek, 2022). | Python has limitations in speed. In a similar way to R, Python can run slow because of poor memory management which can take up computer space. |
| 2 | There is a large Python community and community boards and blogs from which to obtain help. | Python is not strong with mobile computing because it was built to be used in server programming it is rarely used in client side. |
| 3 | Python is easy to read. | Due to slow processing, Python can have runtime errors |
| 4 | It is efficient to write as you need to write fewer lines of code than some other languages | Python (like R) consumes a lot of memory space. |
| 5 | It has a defined structure that makes guidance easy to follow, read, write, and consume | Python is not easy to test because the programmer must clear out each error prior to running (Szkaradek, 2022). |
| 6 | Python enhances productivity because of the ease of which it can be consumed |  |
| 7 | Has dynamic typing so the programmer does not need to declare variables or state datatype while coding |  |
| 8 | Python has powerful integration features which makes it a good choice for building enterprise software applications |  |
| 9 | Python (like R) has a large collection of libraries. Python libraries include machine learning, AI, modeling web and mobile software development |  |
| 10 | With Python programmers do not need to depend on external libraries because it has many functions within the program |  |
| 11 | Python is free and open source |  |
| 12 | Python can be integrated with other platforms |  |
| 13 | Python is an interpreted language meaning it is executed from the source code directly, as long as libraries are linked it can be run within the program. |  |

The pros and cons of Scala programming language are documented in table 3. In summary, Scala is a highly efficient language with less code that needs to be written compared to other languages. This improves efficiency, productivity, and quality (via fewer bugs).

Table 3: Pros and Cons of Scala Programming Language

|  |  |  |
| --- | --- | --- |
| **#** | **Pro** | **Con** |
| 1 | Scala is easy to learn if you know Java as the syntax is similar (Data Flair, 2022a). | The Scala presentation compiler can be slow to load an especially with a relatively large project. Programmers can purchase the Ultimate Intellij package to overcome this. |
| 2 | Scala is a very concise and efficient language to write. For instance, in Java when you need 9 lines of code Scala can be written in 3 (Data Flair, 2022a). | The block selection is difficult to use and changing the workspace loses configurations. |
| 3 | Due to code efficiency this increases productivity. | Scala is not object-oriented programming, (although a developer can switch back), but this can be difficult to conceptualize for developers who have written code in this manner. |
| 4 | Scala has some good IDE’s that are based on Eclipse. It is also possible to customize your own. The best Scala IDE support is in Emacs. | In the IDE’s there may be times when false positives or bugs occur. |
| 5 | The User Interface (UI) for Scala is easy to understand and use. Within the UI the editor is good. | Scala runs on a Java Virtual Machine (JVM) and therefore has no tail-recursive optimization. |
| 6 | Additional features with Scala include Mixins (traits), open classes and monkey patching. With mixings the programmer can carry out multiple inheritances. | Scala (unlike Java, Python and even R) has a limited developer community currently. This is growing, however. |
| 7 | Scala is great for data analysis with support from Spark. |  |
| 8 | A unique advantage of Scala is that it passes functions as arguments to other functions and returns then as values from other functions. This makes Scala a highly functional paradigm. |  |
| 9 | Scala reduces threat safety concerns found in Java by having inherently immutable objects. |  |
| 10 | Scala enables encoding of documents using XML in products. |  |
| 11 | For Java developers Scala is easy to learn as there are many similarities |  |

The pros and cons of Java programming language are documented in table 4. In summary, Java is a very popular language used by programmers (38.4%, (Berkeley Extension, 2022). Java has many advantages compared to drawbacks. As a novice programmer I have found Java more difficult to use than Python as noted in table 4 due to its more verbose and complex syntax.

Table 4: Pros and Cons of Java Programming Language

|  |  |  |
| --- | --- | --- |
| **#** | **Pro** | **Con** |
| 1 | Java is simple to learn and understand especially when compared to C and C++ (Data Flair, 2022b). | Compared to C and C++ Java is slower. However, it is faster when compared to the other Apache Spark languages. |
| 2 | Java is an object-oriented programming language which is an advantage because it allows the programmer to break the program into bite sized chunks (one object at a time) improving productivity and quality | When building User Interfaces (UI) Java is not as good as other languages such as Python |
| 3 | Java is a secure language because unlike C and C++ it does not allow pointers to memory location | Java has no back up features |
| 4 | Java is cheap, easy to maintain | Java requires more memory space than C and C++. However, it is faster when compared Python. |
| 5 | Java is simple to build | Java, compared with Scala and Python especially has complex syntax |
| 6 | Java can run on any machine regardless of the hardware. | Compared with Scala and Python Java is often very verbose |
| 7 | Points 4, 5, and 6 reduce the cost of development due to efficiencies in time. | Java 11 and above requires a commercial license. |
| 8 | The Java platform is independent of the operating system and hardware. This makes its use flexible for use. |  |
| 9 | Java, like Python for example, are high level languages which makes them simpler to program and does not require an interpreter for the computer to understand. |  |
| 10 | Because Java is platform independent and hardware independent it is highly portable and compatible to almost all devices |  |
| 11 | The JVM automatically manages memory and removes objects that do not refer to a class. This makes Java easier to run and faster. |  |
| 12 | Java allows for parallel processing called multi-threading. More than one thread can be run at the same time. This also allows Java to run quickly. |  |
| 13 | Java is a very stable language as it has a huge community of users making bugs identified quickly and updates remove bugs regularly. |  |
| 14 | Java is a distributed language (like R) meaning it can share data and programs across programs. Java has support for a Remote Method Invocation (RMI) which enables distributed processing. |  |

**References**

Berkeley Extension (2022). 11 Most In-Demand Programming Languages in 2022. Retrieved on June 25th, 2022, from: [https://bootcamp.berkeley.edu/blog/](https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/#:~:text=According%20to%20Stack%20Overflow's%202020,(PDF%2C%202.4%20MB)).

Centers for Disease Control (2022). Smoking During Pregnancy. Retrieved on June 25th, 2022 from: [https://www.cdc.gov/](https://www.cdc.gov/tobacco/basic_information/health_effects/pregnancy/index.htm#:~:text=One%20in%20every%20five%20babies,have%20lower%20birth%20weight%20babies).

Chan, C. (2020). What is a ROC curve and how to interpret it. Retrieved on June 25th, 2022,from: <https://www.displayr.com/what-is-a-roc-curve-how-to-interpret-it/>

Data Flair (2022a). Scala Advantages & Disadvantages/Pros and Cons of Scala. Retrieved on June 25th, 2022 from: <https://data-flair.training/blogs/scala-advantages/>

Data Flair (2022b). Pros and Cons of Java/ Advantages and Disadvantages of Java. Retrieved on June 25th, 2022 from: <https://data-flair.training/blogs/pros-and-cons-of-java/>

EMC Educational Services (2015). *Data Science and Big Data Analytics.* Wiley, Inc. Indianapolis, IN.

Erl, T., Khattak, W., Buhler, P. (2016). *Big Data Fundamental: Concepts, Drivers & Techniques.* Prentice Hall. Boston, MA.

Kohavi, R., Wolpert, D.H. (1996). Bias Plus Variance Decomposition for Zero-One Loss Functions. *ICML. 96.*

Hastie, T., Tibshirani, R., Friedman, J. (2017). *The Elements of Statistical Learning*. 2nd Ed. Spring, New York, NY.

Szkaradek, T. (2022). Pros and cons of Python. The Codest: In Code we Trust. Retrieved on June 25th, 2022 from: <https://thecodest.co/blog/pros-and-cons-of-python>

Schaffer, Jonathan (2015). ["What Not to Multiply Without Necessity"](http://www.jonathanschaffer.org/laser.pdf) (PDF). *Australasian Journal of Philosophy*. **93** (4): 644–664.

TechVidvan (2022). *Should you start learning R? Weigh the Pros and Cons of R programming.* Retrieved on June 25th, 2022 from: <https://techvidvan.com/tutorials/pros-and-cons-of-r/>

Triniti (2022). Object Storage vs. HDFS – Which is Better? Retrieved on June 25th, 2022 from: <https://www.triniti.com/data-warehousing-object-storage-vs-hdfs>