**Iowa Liquor Sales Predictive Analysis**

Ankita Paul, Shuvadeep Kundu

Department of Information Systems, California State University

Los Angeles

e-mail: [apaul11@caltstatela.edu](mailto:apaul11@caltstatela.edu), [skundu@calstatela.edu](mailto:skundu@calstatela.edu)

**Abstract:** The objective of our project is to analyze and predict sales of liquor in the state of Iowa by applying machine learning algorithms to models built for prediction. We have taken recourse of Azure ML and Spark ML for our predictive analysis. We have worked on the Iowa liquor sales dataset comprising of records from 2012 to 2019 in 24 columns and approximately 1.8 million rows. We have drawn conclusions by comparing the models with different algorithms applied and their accuracy in predicting the sales using both Azure ML and Spark ML.

**1. Introduction**

Iowa is a state in the Midwestern United States. It is often viewed as a farming state, where agriculture is actually a small portion of the state’s diversified economy. We wanted to look into a different part of Iowa’s economy which is the alcoholic beverage industry. The Iowa Alcoholic Beverages Division is the alcoholic beverage control authority for the U.S. state of [Iowa](https://en.wikipedia.org/wiki/Iowa). Since March 8, 1934, it has regulated the traffic in, and maintained a monopoly on the wholesaling of, [alcoholic beverages](https://en.wikipedia.org/wiki/Alcoholic_beverage) in the state, thus making Iowa an [alcoholic beverage control state](https://en.wikipedia.org/wiki/Alcoholic_beverage_control_state). We wanted to analyze the sales picture of this booming industry and predict the revenue that this industry can generate in future based on its present records.

Our dataset is of size 4.13 GB, has 24 columns and is in CSV format. Since our aim is to predict the sales amount of liquor in Iowa, we have selected the ‘Sales (Dollar)’ column as our label column. Leveraging on our knowledge of machine learning algorithm, we have built and run models to conduct the predictive analysis of Iowa liquor sales.

2. Related Work

Michael [1] and Evan [2] worked on this dataset to build predictive model and conduct predictive analysis but with different goals and techniques than ours.

[1] Michael Salmon built a predictive model using the dataset of liquor sales in Iowa. However, he used a subset of the Iowa liquor sales data which comprised sales records of the year 2015 and the first quarter (Jan – March) of 2016. The objective of the project was to use sales data from 2015 to build a model that could predict total 2016 sales based off Q1 2016 data. The data was analyzed using the Pandas library in Python. This is different from our project because we have used the sales records from 2012 to 2019 to predict sales and have worked on Azure ML and Spark ML.

[2] Evan Lutins conducted predictive analysis by building a linear regression model, however it was only for the rest part of 2016 while we have utilized the records till 2019 for prediction. Also, he used scikit-learning to run the linear regression model and Pandas library to explore the data while we have built and run linear regression models in Azure ML and Spark ML

3. Hardware Specifications

For this project, we have used Microsoft Azure Machine Learning Studio for Azure ML algorithms and Databricks Community Edition to implement Spark ML algorithms. We have also used Hadoop cluster on the Oracle Big Data Cloud platform to run PySpark commands for the predictive analysis. The specifications are as follows:

|  |  |  |
| --- | --- | --- |
| **Azure** | **Databricks** | **Oracle BDCE \*** |
| * **Memory** – 10 GB * **Nodes** - 1 * **Max no. of modules per experiment** -   100 | * **Memory** – 15.3 GB * **Nodes**- 1 * **Driver -** (2 cores, 1 DBU) * **Databricks Runtime Version** – 6.5 (Scala 2.11, Spark 2.4.5) * **Python version** - 3 | * **Memory** – 247.625312 GB * **Storage** – 1003.6 GB * **Nodes** – 2 * **No. of processors** – 32 * **Cluster version** - Hadoop 2.7.1.2.4.2.0-258 * **CPU speed** – 2.20 Ghz * **Python version** – 2.7.14 * **Spark version** – 2.1.0.2.6.0.3-10 |

**Table 1. Hardware specifications**

4. Background/Existing Work

The models that we have built in our projects are mostly based on our existing works. We have used Machine Learning algorithms in both Azure ML and Spark ML.

**4.1 Regression**

The Linear Regression, Decision Forest Regression and the Boosted Decision Tree Regression models are based on the study of the prediction of heating load in Azure ML Studio. We had used Linear Regression and Decision Forest Regression model in the prediction of heating load for the energy efficiency dataset. Cross-validate module and Tune Model Hyperparameters module were used for training the model. Permutation Feature Importance module was used to check the scores of importance of the features on the basis of which, features were pruned to improve the performance.

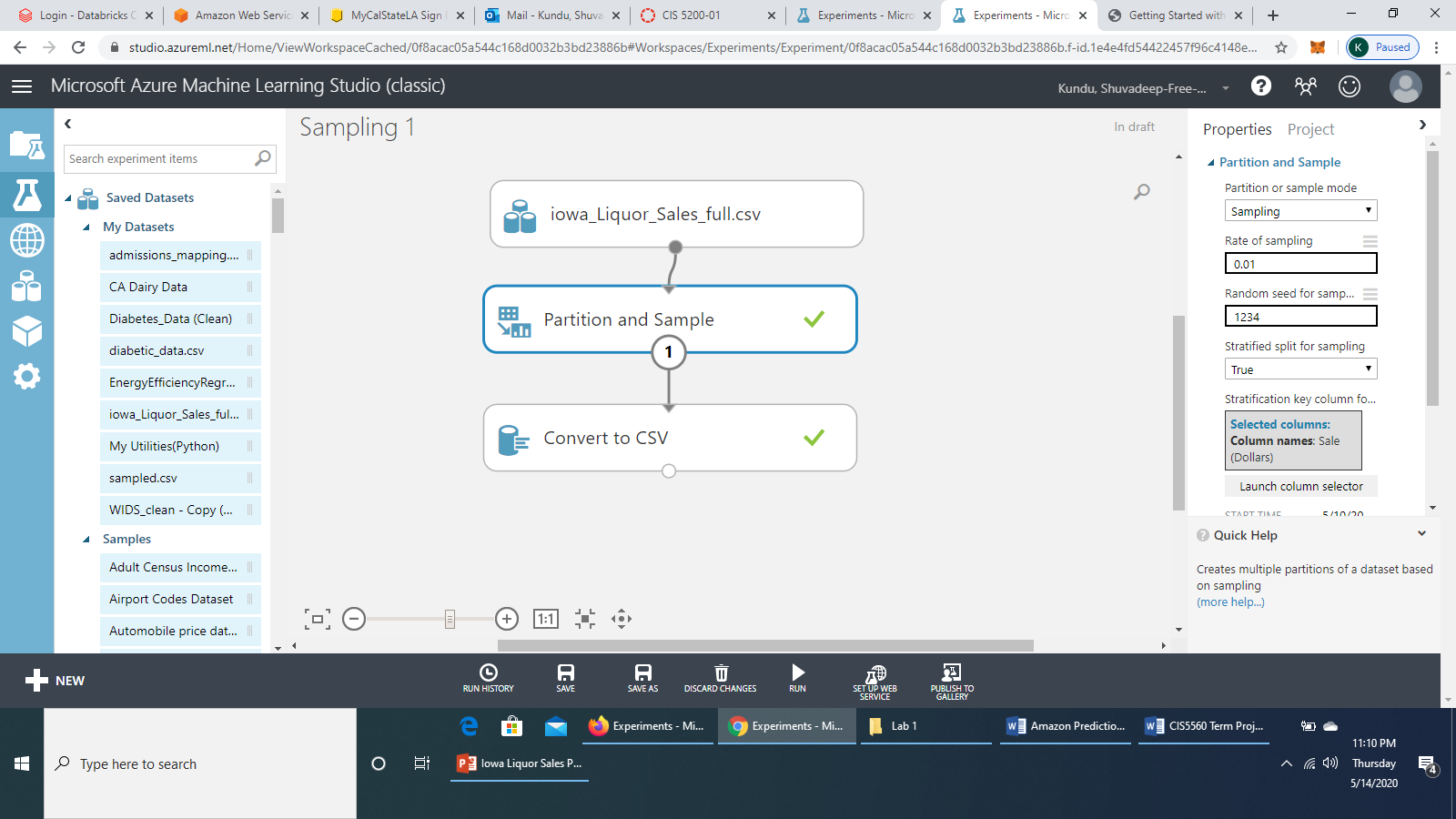
In Spark ML, we used Linear Regression model using both CrossValidator as well as TrainValidationSplit. This is based on the lab where we had predicted arrival delay of the flights using Linear regression, where CrossValidator was used for training purpose. We have also used Decision Tree Regression and Gradient Boosted Tree Regression in Spark ML which were not done previously in the lab, but the application is similar to Linear Regression model. RMSE was used as an evaluation parameter.

5. Our Work

We built models in both Azure ML and Spark ML to predict the “Sales (Dollars)” in Azure ML and “SaleInDollars” in Spark ML.

**5.1 Azure ML**

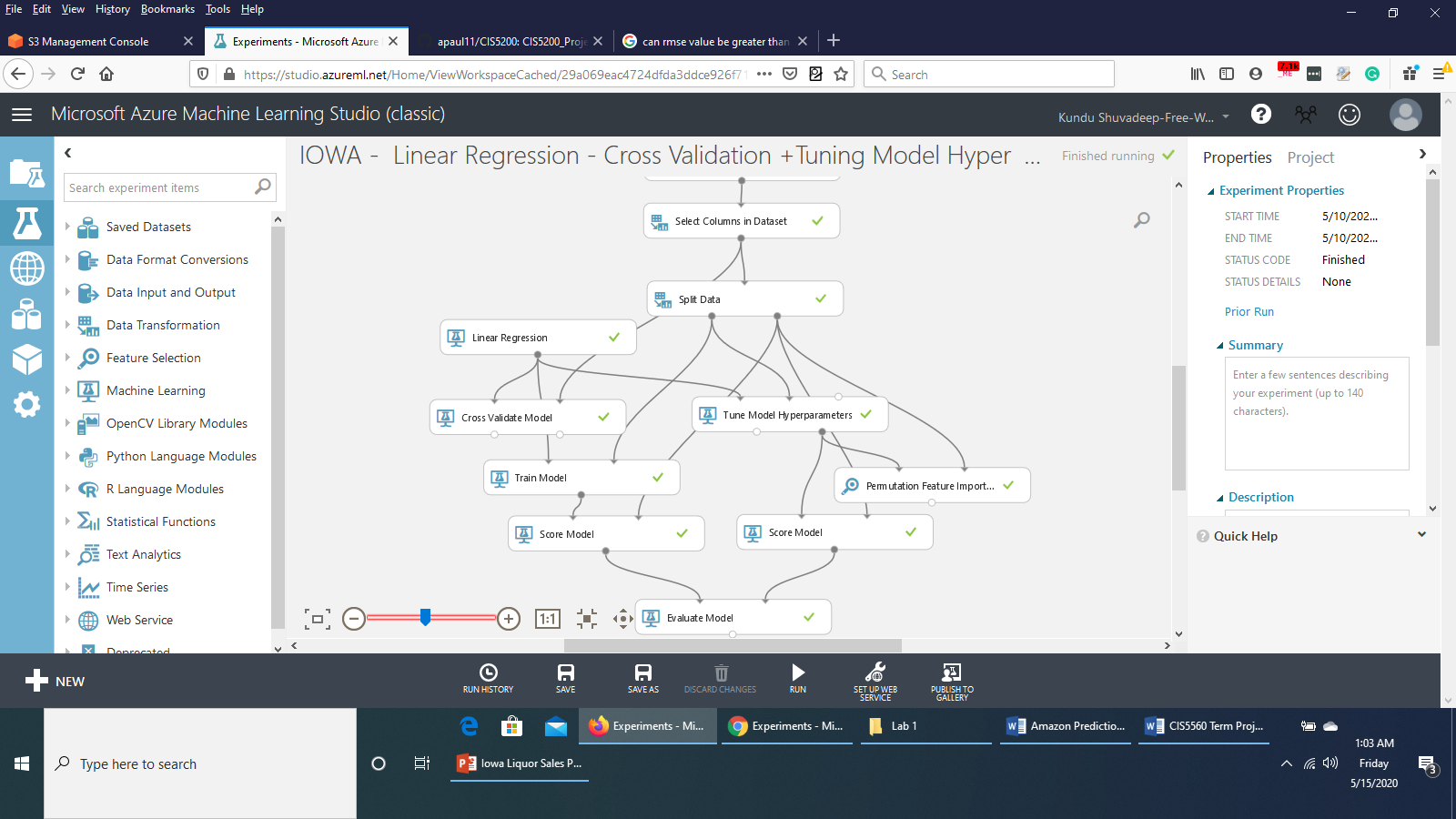
Due to storage issue in Azure ML, we sampled the original dataset of 4.13 GB to 6.45 MB on Azure ML Studio using Partition and Sample module with stratified sampling set to True by selecting the “Sale (Dollars)” column to ensure that the sampled dataset is a true representative of the original dataset. However, the experiments were failing due to memory exhaustion, so we further sampled the 6.45 MB dataset at the rate of 0.095 and used it for our experiments.



**Figure 1. Sampling of original dataset at 0.01**

**5.1.1 Linear Regression (Azure ML)**

We selected all columns, all features (the sampled dataset contained 17 columns) and selected Sale (Dollars) as label column. After splitting the dataset at 70:30 train – test ratio using Split Data module, we used Cross Validate Model and Tune Model Hyperparameters to train the model. We selected RMSE as metric for measuring performance and had a run time of 37.24 minutes.



**Figure 2. Linear Regression**

We also used Permutation Features Importance module to check the important features affecting the performance of the model.

|  |  |
| --- | --- |
| **Features** | **Scores** |
| Bottles Sold | 380.17 |
| Volume Sold (Liter) | 77.257 |
| Pack | 66.775 |
| Vendor Number | 62.78 |
| Category | 51.802 |
| Bottle Volume (ml) | 44.811 |
| Item Number | 19.236 |
| State Bottle Cost | 12.448 |
| State Bottle Retail | 9.823 |
| City | 1.0707 |
| County Number | 0.924 |
| Zip Code | 0.777 |
| Date | 0.243 |
| Invoice/Item Number | 0 |
| Store Number | -1.846 |
| Store Location | -1.9402 |

**Table 2. Feature Importance table (Linear Regression)**

Both the Cross Validate Model and Tune Model Hyperparameters performed equally well.

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Cross Validation** | **Tune Model Hyperparameters** |
| **RMSE** | 137.145669 | 137.145669 |
| **Coefficient Of Determination** | 0.925955 | 0.925955 |

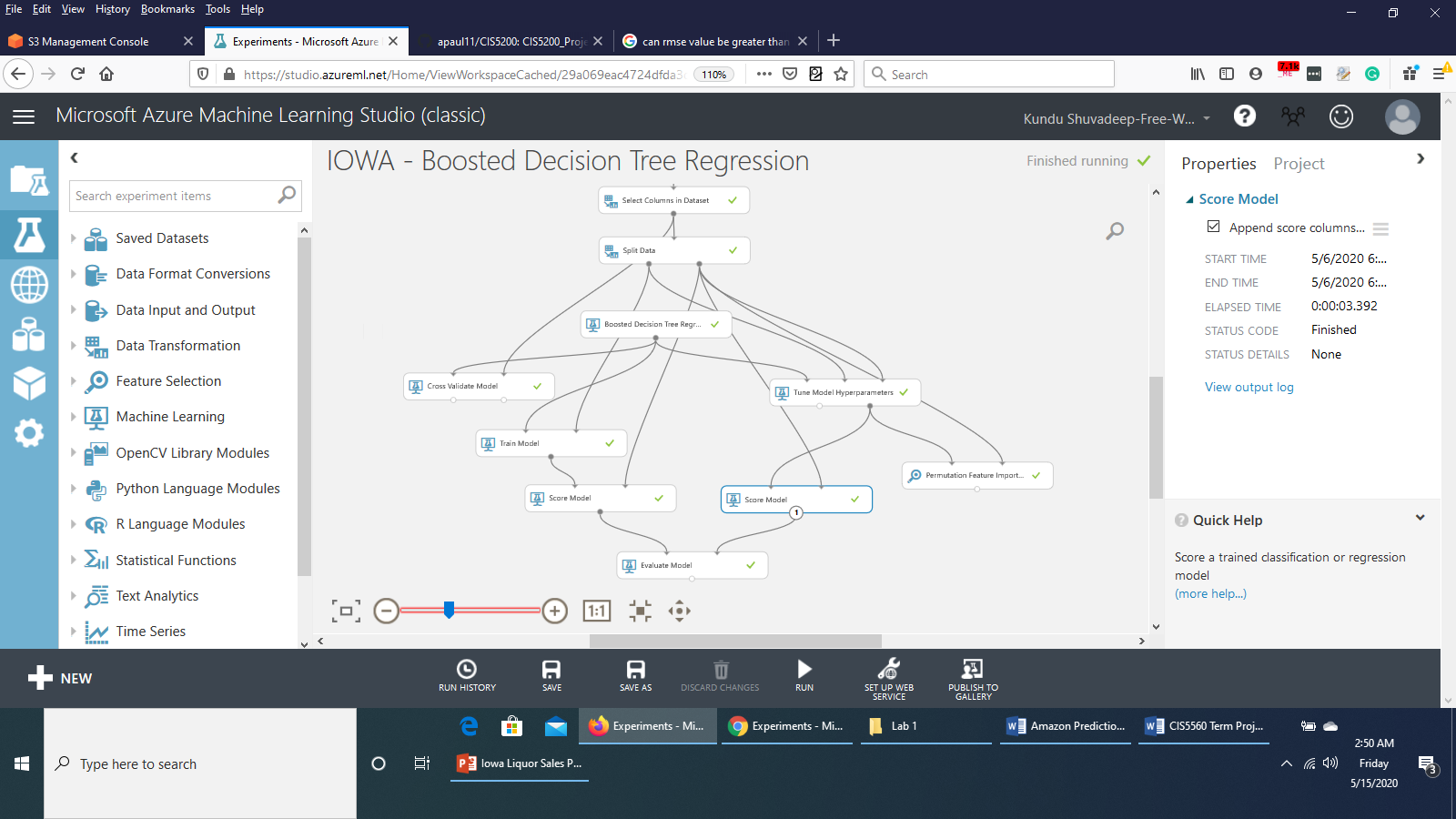
**Table 3. Evaluation results of Linear Regression (Azure ML)**

Pruning features Invoice/Item number, Store Number and Store Location resulted in a higher RMSE value of 240.403611 and lower Coefficient of Determination value of 0.772484.

Hence the model before pruning the features gave better accuracy for our prediction.

**5.1.2 Boosted Decision Tree Regression (Azure ML)**

We selected all columns, all features from the sampled dataset and selected “Sale (Dollars)” as label column. We split the data at 70:30 ratio for training and testing. We chose Single Parameter as Create trainer mode, Maximum number of leaves per tree as 20, Total number of trees constructed as 100. We used Cross Validate Model, Tune Model Hyperparameters and Permutation Features Importance modules. We selected RMSE as metric for measuring performance and it had a run time of 2.17 minutes.



**Figure 3. Boosted Decision Tree Regression**

The Tune Model Hyperparameter performed better than Cross validation, with lower RMSEvalue and higher Coefficient of Determination value.

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Cross Validation** | **Tune Model Hyperparameters** |
| **RMSE** | 267.974043 | 173.140216 |
| **Coefficient Of Determination** | 0.717307 | 0.881988 |

**Table 4.** **Evaluation results of Boosted Decision Tree Regression (Azure ML)**

Having a look at the Features Importance table, we could find out the features that affected the performance of the model the most and the features that had least importance.

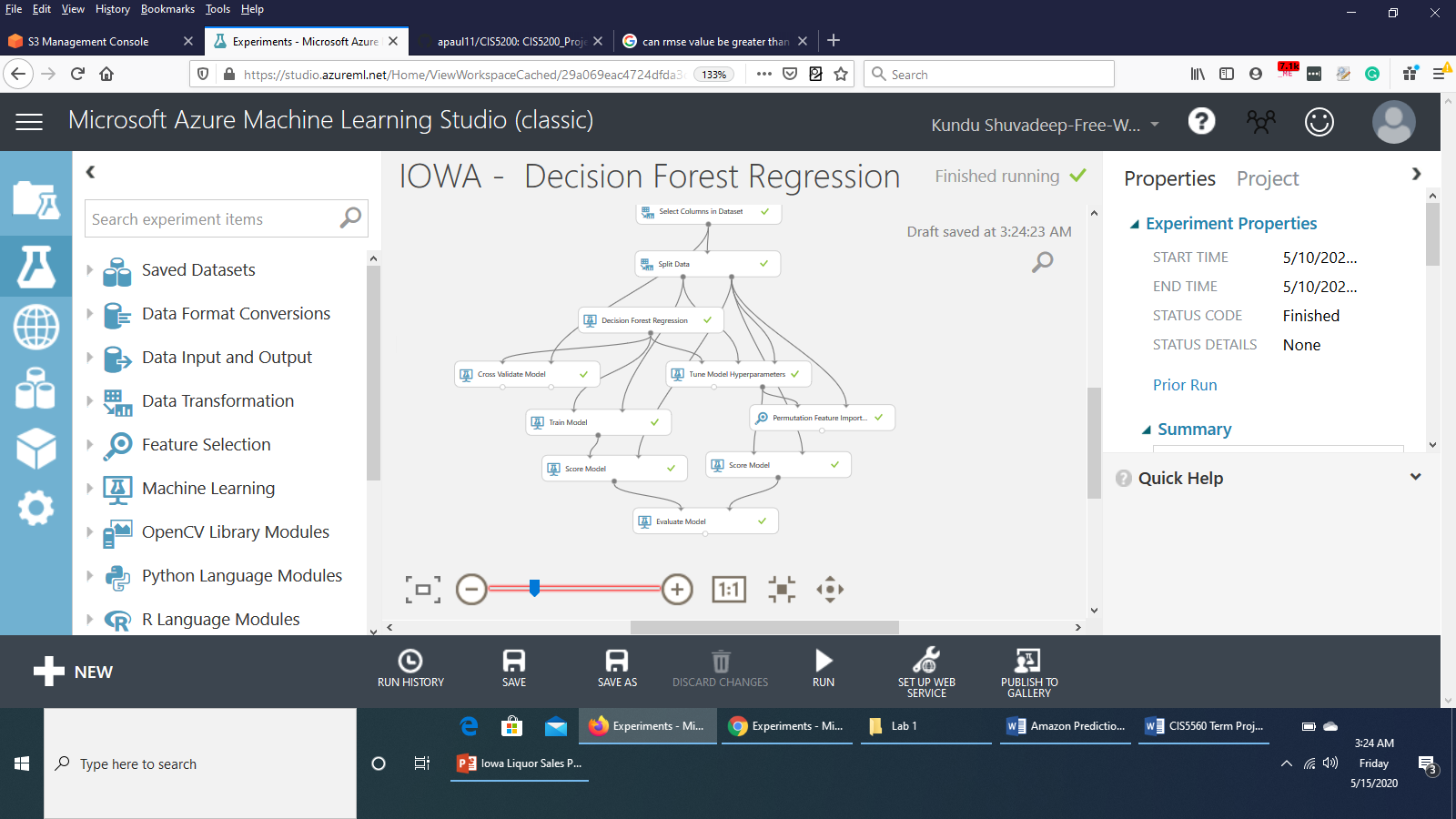
|  |  |
| --- | --- |
| **Features** | **Scores** |
| Bottles Sold | 454.818329 |
| Pack | 81.371853 |
| Vendor Number | 75.608935 |
| Category | 54.132026 |
| Bottle Volume (ml) | 39.98075 |
| Volume Sold (Liters) | 6.885084 |
| State Bottle Cost | 2.324521 |
| Store Number | 0.948541 |
| City | 0.295601 |
| Item Number | 0.219255 |
| Zip Code | 0.167606 |
| State Bottle Retail | 0.015421 |
| Invoice/Item Number | 0 |
| Date | 0 |
| Store Location | 0 |
| County Number | -0.122389 |

**Table 5. Feature Importance table (Boosted Decision Tree Regression)**

Pruning the less important features Invoice/Item Number, Date, Store Location, County Number, and Zip Code led to decrease in RMSE value by very negligible amount.

**5.1.3 Decision Forest Regression (Azure ML)**

We selected all columns, all features from the sampled dataset, selected “Sale (Dollars)” as label column and split the data in 70:30 ratio for training and testing. In the Decision Forest Regression module, we chose Bagging as Resampling Method, Single Parameter as Create trainer mode, Number of decision trees as 8, Maximum depth of the decision trees as 32, Number of random splits per node as 128 and Minimum number of samples per leaf node as 4. We used Cross Validation, Tune Model Hyperparameters, Permutation Feature Importance modules and the experiment had a run time of 36.79 seconds.



**Figure 5. Decision Forest Regression**

Evaluation results revealed that Tune Model Hyperparameter performed better with lower RMSE value and higher Coefficient of Determination than Cross Validate Model.

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Cross Validation** | **Tune Model Hyperparameters** |
| **RMSE** | 338.365264 | 314.118924 |
| **Coefficient Of Determination** | 0.549286 | 0.611566 |

**Table 6.** **Evaluation results of Decision Forest Regression (Azure ML)**

The scores of feature importance are as follows:

|  |  |
| --- | --- |
| **Features** | **Scores** |
| Bottles Sold | 168.902577 |
| Volume Sold (Liters) | 45.997167 |
| Bottle Volume (ml) | 27.05422 |
| Pack | 15.378501 |
| State Bottle Cost | 2.951751 |
| Vendor Number | 2.912818 |
| County Number | 1.690925 |
| Category | 1.339969 |
| Item Number | 1.106864 |
| State Bottle Retail | 0.719819 |
| City | 0.201144 |
| Invoice/Item Number | 0 |
| Date | 0 |
| Zip Code | -0.030835 |
| Store Location | -0.033807 |
| Store Number | -0.26712 |

**Table 7. Feature Importance table (Decision Forest Regression)**

Excluding the columns Store Number, Store Location, Zip Code, Date, Invoice/Item Number resulted in considerably decreased RMSE value and increased Coefficient of Determination value. However, Cross Validation Model performed better this time.

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Cross Validation** | **Tune Model Hyperparameters** |
| **RMSE** | 246.037976 | 271.932532 |
| **Coefficient Of Determination** | 0.761695 | 0.708894 |

**Table 8.** **Evaluation results of Decision Forest Regression after pruning features (Azure ML)**

**5.2 SparkML**

We worked on both Databricks Community Edition and Oracle BDCE to run our experiments using Spark ML algorithms.

In Databricks CE, we used a 41.1 MB sized sample dataset, imported it as a table after creating a Test Cluster in Databricks and ran the experiments on the records of that table.

In Oracle BDCE, due to memory exhaustion issue, we could not run the experiment on the original big dataset but used a sample dataset of size 238 MB.

**5.2.1 Linear Regression (Databricks CE - Spark ML)**

To predict the sales amount of Iowa liquor sales with Linear Regression algorithm, we used the columns with Integer and Double data types and cast them all into Double data type. The features used were Pack, BottleVolumeInMl, StateBottleCost, StateBottleRetail, BottlesSold, VolumeSoldInLiters and the label column selected was “SaleInDollars” since we are going to predict the sales amount. We split the data into 70:30 ratio of train and test datasets. Vector Assembler was used to assemble the features in a vector. LinearRegression was used and pipeline was defined. In the Parameter Grid, the parameters regParam and maxIter were defined. We used both CrossValidator (with number of folds as 10) and TrainValidationSplit (with train ratio of 0.8) to train the model. Pipeline was used as an estimator and run with fit() method on training dataset. RegresionEvaluator was used in both cases to retrieve the RMSE value and each of the models tested with Cross Validation and TrainValidationSplit gave RMSE value of 141.96804. The runtime was 4.07 minutes.

**5.2.2 Gradient Boosted Tree Regression (Databricks CE - Spark ML)**

In Gradient Boosted Tree Regression algorithm, we used the features Pack, BottleVolumeInMl, StateBottleCost, StateBottleRetail, BottlesSold, VolumeSoldInLiters and the label column as “SaleInDollars”, all cast into Double data types. We split the data into 70:30 ratio of train and test datasets. Vector Assembler was used to assemble the features in a vector. GBTRegressor was used and Pipeline was defined. In the Parameter Grid, the parameters maxDepth, minInfoGain, and stepSize were defined. TrainValidation Split method with train ratio of 0.8 was used. The pipeline was used as an estimator and was run with fit() method on training dataset to train the model. RegressionEvaluator was used as an evaluator to retrieve the RMSE value which was 115.3823. The whole experiment took 4.14 minutes to run.

**5.2.3 Decision Tree Regression (Databricks CE - Spark ML)**

In Decision Tree Regression algorithm, we used the features Pack, BottleVolumeInMl, StateBottleCost, StateBottleRetail, BottlesSold, VolumeSoldInLiters and the label column as “SaleInDollars”, all cast into Double data types. We split the data into 70:30 ratio of train and test datasets. Vector Assembler was used to assemble the features in a vector. DecisionTreeRegressor was used and Pipeline was defined. ParamGridBuilder was used. CrossValidator was used for training with number of folds as 5. The pipeline was used as an estimator and was run with fit() method on training dataset to train the model.

RegressionEvaluator was used as an evaluator to retrieve the RMSE value which was 94.73676. The experiment took 1.86 minutes to run.

**5.2.4 Linear Regression (Oracle BDCE – Spark ML)**

We used Linear Regression to predict sale amounts. The features used were Pack, BottleVolumeInMl, StateBottleCost, StateBottleRetail, BottlesSold, VolumeSoldInLiters and the label column was “SaleInDollars”, all cast into Double data types. We split the data into 70:30 ratio of train and test datasets. Vector Assembler, LinearRegression was used and pipeline was defined. Both Cross Validation and TrainValidationSplit was used and RegressionEvaluator was used to retrieve RMSE. The model with CrossValidator took 4.15 minutes to run and gave a RMSE value of 182.1768. The model with TrainValidationSplit performed a bit better with lower runtime of 45 seconds and bit lower RMSE value of 181.1703.

**5.2.5 Gradient Boosted Tree Regression (Oracle BDCE – Spark ML)**

In Gradient Boosted Tree Regression algorithm, we used the features Pack, BottleVolumeInMl, StateBottleCost, StateBottleRetail, BottlesSold, VolumeSoldInLiters and the label column as “SaleInDollars”, all cast into Double data types. We split the data into 70:30 ratio of train and test datasets. Vector Assembler, GBTRegressor was used and pipeline was defined. The parameters maxDepth, minInfoGain, and stepSize were defined in the Parameter Grid. TrainValidation Split method with train ratio of 0.8 was used. RegressionEvaluator retrieved RMSE value of 170.163943 and the whole model took 3.23 minutes to run.

**5.2.6 Decision Tree Regression (Oracle BDCE – Spark ML)**

In Gradient Boosted Tree Regression algorithm, we used the features Pack, BottleVolumeInMl, StateBottleCost, StateBottleRetail, BottlesSold, VolumeSoldInLiters and the label column as “SaleInDollars”, all cast into Double data types. We split the data into 70:30 ratio of train and test datasets. Vector Assembler, DecisionTreeRegressor was used and pipeline was defined. CrossValidator was used for training with number of folds as 5. RegressionEvaluator retrieved RMSE value of 116.422847 and the whole model took 20 seconds to run.

**6. Conclusion**

From all our experiments, we can summarize that:

1. Linear Regression performed better in Spark ML than in Azure ML. While in Azure ML, after pruning features, the RMSE value for Linear Regression had increased; in Spark ML, pruning some more features led to lower RMSE value than in Azure ML.

2. In Azure ML, the model with Linear Regression performed best with lowest RMSE value and highest Coefficient of Determination.

3. In Spark ML, the model with Decision Tree Regression performed best with lowest RMSE value as well as lowest run time.

4. We faced storage issue with big data in PySpark CLI both with the original 4.13 GB sized data and a sampled data of 986 MB. Finally we could run the commands with a smaller sampled data of 238 MB on PySpark CLI.

We can have a better picture of our summary from the following comparison tables for Azure ML and Spark ML.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metrics** | **Linear Regression** | **Boosted Decision Tree Regression** | **Decision Forest Regression** |
| **RMSE** | 137.14566 | 171.97299 | 246.03797 |
| **Coefficient of Determination** | 0.925955 | 0.883574 | 0.761695 |
| **Run time** | 37:24 minutes | 2.17 minutes | 36.798 seconds |

**Table 9. Comparison of Models in Azure ML**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metrics** | **Linear**  **Regression** | **Gradient**  **Boosted**  **Tree**  **Regression** | **Decision Tree**  **Regression** |
| **Databricks CE** | | | |
| **RMSE** | 141.9680 | 115.382 | 94.736 |
| **Run time** | 4.07 mins | 4.14 mins | 1.86 mins |
| **PySpark CLI** | | | |
| **RMSE** | 181.17  (TVS) | 170.164 | 116.42 |
| **Run time** | 45 sec | 3.23 mins | 20 sec |

**Table 10. Comparison of Models in Spark ML**

### References

[1] Michael Salmon, “Predictive Modeling with Iowa State Liquor Sales Data”, *towards data science,* 2017 [Online]. Available on : https://towardsdatascience.com/predictive-modeling-with-iowa-state-liquor-sales-data-e45342081b83

[2] Evan Lutins, "Predicting-Iowa-Liquor- Sales", GitHub, 2017[Online]Available on: https://github.com/elutins/Predicting-Iowa-Liquor-Sales

[3] Microsoft's DAT203x, Data Science and Machine Learning Essentials

[4] Data Science and Machine Learning Essentials: Lab 2A – Acquiring Data in Azure Machine Learning.

Online link:

https://calstatela.instructure.com/files/2788092/download?download\_frd=1.

[5] Data Science and Machine Learning Essentials: Lab 4b – Modeling with Feature Pruning.

Online link:

https://calstatela.instructure.com/files/2788119/download?download\_frd=1

[6] https://turi.com/learn/userguide/supervised-learning/boosted\_trees\_regression.html

[7] https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#gradient-boosted-tree-regression

[8] Original dataset path link:

https://projectcis5200.s3-us-west-1.amazonaws.com/iowa\_Liquor\_Sales.csv

[9] Sampled dataset path link for PySpark CLI (238 MB):

https://project5560.s3-us-west-1.amazonaws.com/Iowa\_Liquor\_Sales\_Cleaned\_sample.csv

[10] GitHub link:

https://github.com/apaul11/CIS5560