1. Author: Hannah Khuong

2. Use Case Name:

Using Spark MLib in Machine Learning, predict survival in Titanic dataset

3. Introduction:

The goal of this project is to predict the survival rate of passenger on the Titanic. This project used multiple methods to predict the survival rate, such as regression, decision tree and random forest. Basic EDA and data manipulation to get the data ready for machine learning were also done. I added few more lines of code about model evaluation to better understand the effectiveness of models.

4. Dataset Used: Titanic dataset (https://www.kaggle.com/c/titanic/data)

This dataset contains 1309 information of passengers on this historic Titanic ship. Each row is the information of a passenger, such as name, age, gender, airfare, survival status (training dataset).

The provided dataset was split into training (891 observations) and testing (418 observations) datasets already. However, I combined these 2 datasets into one titanic.csv (1309 observations in total) and split them into training, testing datasets later in the code.

5. Technical Details:

This project uses MLib library, Spark SQL, and Spark Dataframe developed by Spark. First of all, I created a SparkSession. This is an entry point to let Spark programming communicate with Dataframe and Dataset API. From spark.sql module, I also imported different built-in functions for dataframe. They are functions regarding basic statistic (i.e., mean, max), regular expression (regexp_extract), and dataframe manipulation (i.e., col, when).

In this library, I used the Spark ML Pipeline, StringIndexer, VectorAssembler, Classification, and MultiClassClassificationEvaluator. I also added code that imports metrics from Scikit-learn to create confusion matrix. Spark Pipeline makes it easy for users to preprocessing dataframe by creating stages of the pipeline, fit, transform data and estimate in model as the end of the pipeline. I used StringIndexer to convert categorical values to numeric values. StringIndexer is a convenient function for machine learning task to quickly convert string into number. This function is essential for models that takes only numbers for calculation, such as support vector machine, Naïve Bayes, decision tree, etc. The VectorAssembler is used to put every variable into a vector called "features". This method of preprocessing predictors is different from normal Python, in which can take the whole dataframe as predicting factor.

To call machine learning API and algorithms for model creation, I had to import them from spark.ml.classification. This module has algorithm for multiple machine learning methods as described in the results. Lastly, the evaluate function was called from the MultiClassClassificationEvaluator to calculate accuracy and f-score.

6. Results:

In the tragic Titanic event, only 37.7% passengers survived. The survival rate in gender is disparate, 82.6% women survived while only 12.0% men survived. The survival rate is also distinct between passenger class. In class 1, 57.5% of passengers lived, 42.2% of passengers in class 2 lived, and only 26.9% passengers survived the tragedy.

Of all machine learning model, *Naïve Bayes* produced the worst accuracy and f-score. This result makes sense because this data has a high level of multicollinearity. For example, Pclass and Fare can be highly correlated, SibSP and FamilySize definitely correlates with each other. Dealing with data that has multicollinearity really contaminates the calculation of Naïve Bayes.

Random forest was the best-performing model. It was no surprise that it out-performed the *decision tree* because random forest usually corrects the overfitting disadvantage of decision tree. Gradient-boosting decision tree, however, performed worse than decision tree. It is due to this noisy data, especially unscaled, makes gradient-boosting machine even more sensitive to noise that leads to over-fitting.

Support vector machine is only available with linear model in PySpark. This is a limitation to improve the results because there are factors in the predictors that might be described better with nonlinear model (i.e., age in quadratic form).

A limitation for *logistic regression* models is that categorical variables as being used similarly as continuous variables. The VectorAssembler only takes numeric numbers and does not take in string. That being said, the features for logistic regression were not ideal for this model. Categorical items should be kept as discrete variables, such as "Sex" and "Alone" variable for accurate training and result interpretation.

Another limitation for these models was that there was no scaling. For models that are sensitive to distance, such as support vector machine, scaling is important for accuracy. With unscaled data, the different ranges of variables can skew the calculation. One variable can overwhelm the overall computation is they are disproportionately greater than other variables. In our case, "Fare" has much greater scale than family size and can introduce bias to support vector machine model.

7. Insight

The prediction of survival can be helpful to identify people at risk in a disaster, man-made or natural. Titanic accidence is just an incidence that makes patterns of behaviors become obvious at the moment of life and death. In a disaster, the unprivileged are often people who suffer the negative consequences of an unfortunate event. For example, as global warming is happening, people who are at higher risk of global warming consequences (i.e., flood, heat wave) are poor people with minimum physical support in such unfortunate events. By using machine learning to mine the pattern of who are at higher risk, policy makers can provide appropriate preventions to reduce the severity of the catastrophe. Prediction itself is a good way to anticipate in an event of disaster to have appropriate measures of preparation. Furthermore, model can be understood to identify the at-risk groups to specifically help them. This action can be done by studying the coefficient and odds ratios in logistic regression model or analyzing decision tree.

8. Debugging Detail

a. In RandomForestClassifier, they incorrectly call DecisionTreeClassifier function. I corrected with RandomForestClassifier function.

b. I merged training and testing data into a titanic.csv. My EDA is different from the instruction. I hold out testing and training data later in the program. I have to add the treatment for missing value in fare (after merging data) by importing avg from module pyspark.sql.functions.

c. Instead of importing data from an AWS bucket, I imported the csv file by uploading that file to the folder.

d. I used toPandas() function to display dataset nicer (titanic_df.limit(5).to Pandas().head()). This is helpful especially after I added 2 more columns.

e. I added confusion matrix, and calculation of f-score for better understanding of the models. I wanted to calculate precision and recall too. However, precision and recall param did not work in MulticlassClassificationEvaluator. I can only add f-score calculation.

9. Conceptual Framework:

The framework for this project is machine learning, which was discussed in the seventh class. Machine learning is to give computers the ability to learn. The machine can make decision without specific coded behaviors programmers give. At its core, machine learning is constructed by sets of algorithms to process data. There are three main types of training: unsupervised learning, supervised learning, and reinforced learning. Machine learning has various application in our world nowadays, from finance, healthcare to marketplace to identify scam, diagnose disease or cluster customers. Spark is a good platform for machine learning. Since machine learning is computationally heavy, the in-memory processing of Spark makes this process faster and allows for stream-processing. It is scalable, and also simple and compatible. The Spark MLib library also provides data engineers and scientists with familiar and popular tools, such as R and Python. Because Spark MLib is built based on these popular languages, they also inherit large number of packages and modules that users have been used to using, such as scikit-learn and statsmodule.

10. Five Vs

Volume	Lets' take an example where we want to build a model to predict the mortality in one of multiple potential disasters. Our model can combine data from all the disaster events in the past across the world. The number of rows is high. The number of features (columns) can also be high, as it will include various individual information and event-related information. This amount of data is definitely high in volume.
Velocity	As a new even occur, we can insert more data in the system. The disaster-related information is not that frequent as other kind of big data, such as click-stream data. Last year (2018), the total number of natural disasters of all the world was below 300. Also, the data for this kind of project do not move around that much. After being pre-processing, it can stay in a database.
Variety	Multiple types of data can be used, such as geology data (earthquake), weather data (flood, drought), people data (address, living condition), event data (narrative, photo of the event, news related to the event). While weather/geology data can be numeric, people-information and narrative data are text. News for information about the event can be text, photo, video, or audio.
Veracity	The data about geology, weather can be highly accurate because institutes studying these matters can provide such data for disaster events. However, information about a man-made disaster (shooting) might need more caution in gathering data. Similarly, information about victims/survivors, narratives of the event are more susceptible to errors.
Value	In this project, I was able to extract patterns underlying data. They are the relationship between survival rate and gender, relationship between survival rate and passenger class. In a bigger prospective of this project, we can derive patterns of people who are at higher risk in a catastrophic event to have preventive methods before it even happens.

11. References

https://spark.apache.org/docs/1.2.2/ml-guide.html Spark pipeline
https://www.datacamp.com/community/tutorials/apache-spark-tutorial-machine-learning
https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80 Gradient-boosting machine
https://spark.apache.org/docs/2.2.0/api/python/pyspark.ml.html#pyspark.ml.classification.LogisticRegression Spark ML API
https://www.infoworld.com/article/3031690/why-you-should-use-spark-for-machine-learning.html Spark ML
Lecture 5 – BigData Platform
Lecture 1 – Big Data
Lecture 7 – Machine Learning
https://ourworldindata.org/natural-disasters Data on natural disaster.

12. A list with short descriptions of all fields in dataset used

survival - Survival (0 = No; 1 = Yes)
class - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
name - Name
sex - Sex
age - Age
sibsp - Number of Siblings/Spouses Aboard
parch - Number of Parents/Children Aboard
ticket - Ticket Number
fare - Passenger Fare
cabin - Cabin
embarked - Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
boat - Lifeboat (if survived)
body - Body number (if did not survive and body was recovered)

13. Appendix

Data treatment:

Missing data treatment:

- Age: replaced by average age of each title initial (Mr, Mrs, etc.)
- Initials are separated into a column and normalized with only Mr, Mrs, Miss and Other.
- Port of Embarkment: replaced with mode
- Cabin: Dropped
- Fare: replaced missing values with mean of fare.

New columns to understand effect of family size

- Created new column named “family size” as sum of Parch(parents/children) and SibSp(siblings/spouses)
- Created new column named “Alone”, which people have family size 0 marked as 1 (as in “yes, they were by themselves on the boat”)

Variable type manipulation

- Converted to numeric and put into an array to be ready for machine learning process.

Machine learning evaluation results

	Logit	**Random Forest**	**Decision Tree**	**Gradient-boosted Tree**	**Naïve Bayes**	**Support Vector Machine**
Accuracy	86.1%	86.9%	86.1%	85.3%	68.9%	85.3%
F-score	85.9%	86.6%	85.9%	85.1%	67.7%	85%