CS 5304 Data Science In The Wild

Homework 1 - Titanic

Weiming Zhang

I. Objective

The goal of this assignment is to generate the equivalent of a Data Science "Hello World" application for Spark 2.0.2. This is your first practice for proving to the Instructor Team that you are capable of building a Data Analysis pipeline for Spark.

II. Problem 1: Titanic Survivors Analysis

a) *Answer the question: "for subgroups of people boarding the Titanic, how would you maximize their individual probability of survival?". You must define meaningful subgroups. You should submit your predictions in a file that clearly labels identity of person and the prediction.*

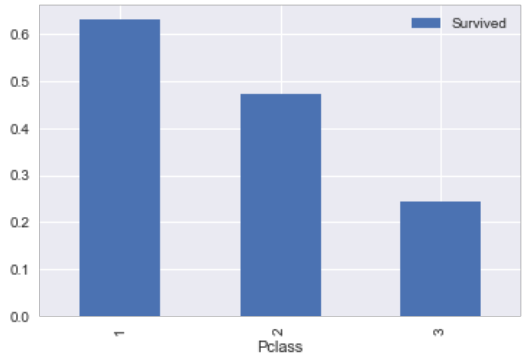
Subgroup: *[Pclass, Title, Sex, AgeRange, PartySize, FareRange, Embarked]*

Pclass	1
Title	Mrs
Sex	Female
AgeRange	Very young
PartySize	2 – 4
FareRange	As expensive as possible
Embarked	C

The raw data columns:
[PassengerID, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked]

Step 1 Drop Unrelated Columns From Features
Drop PassengerID, Survived, Ticket, Cabin from features since they are either irrelevant with the result or varies very much across individuals.
[Pclass, Name, Sex, Age, SibSp, Parch, Fare, Embarked]

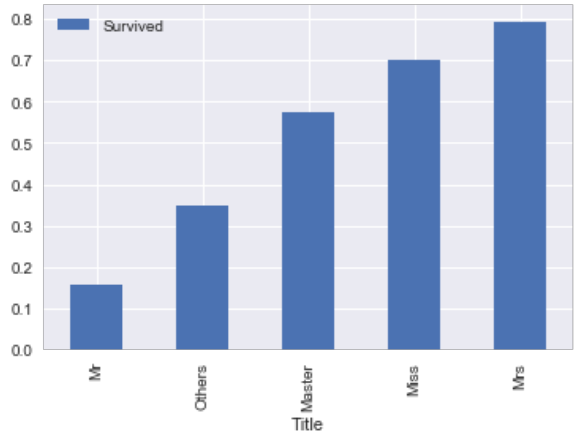
Step 2 PClass



We see that chance of survival P(1) > P(2) > P(3)

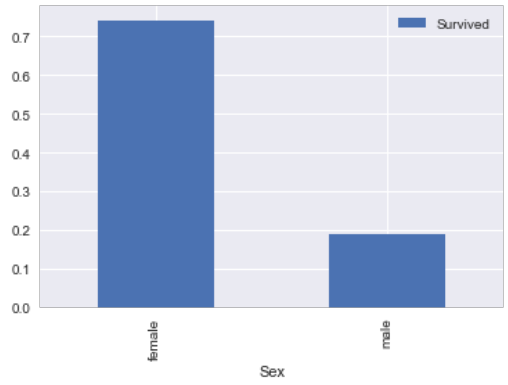
Step 3 Name
The name as a whole does not indicate chance of survival, however, the title of the name would indicate gender and age in some way. So I extracted the titles from the names:

	Male	Female
Miss		185
Mrs		125
Mr	517	
Master	40	
Others	3	20



There is a correlation between the title and chance of survival. We can drop name and use title instead for our features.
[Pclass, Title, Sex, Age, SibSp, Parch, Fare, Embarked]

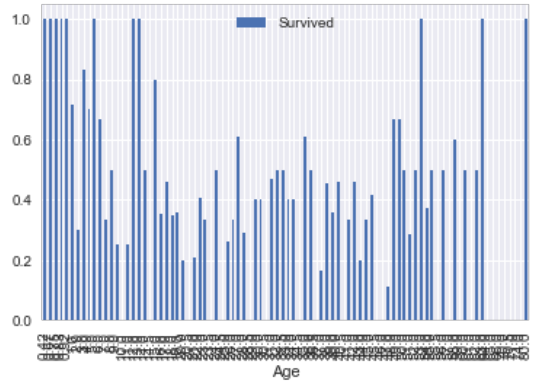
Step 4 Sex



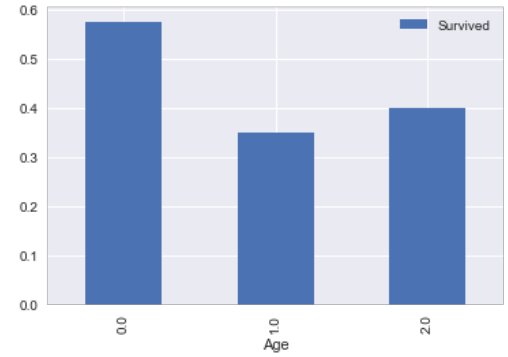
Females have higher chance of survival than males. We can keep sex as a feature.

Step 6 Age

Age plays an important role in determine survival as well, however it spreads out so much across passengers. After some tweaking, we found that very young passengers have a higher chance of survival than older ones.

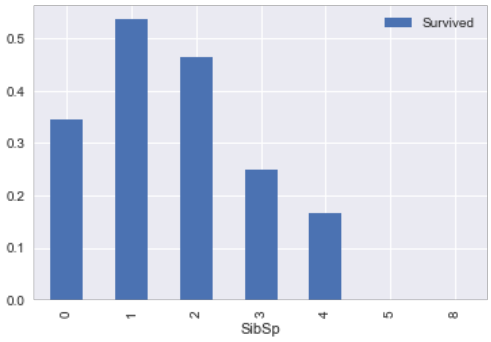


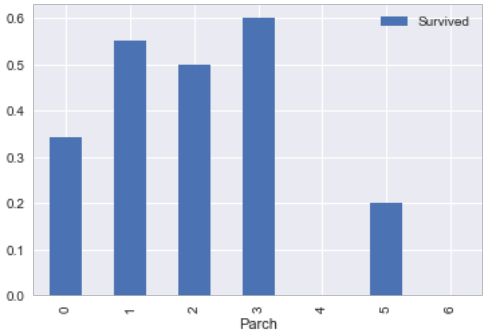
After some tweaking, I decide to divide age into three ranges:
[0-15], [15-35], [35-80]



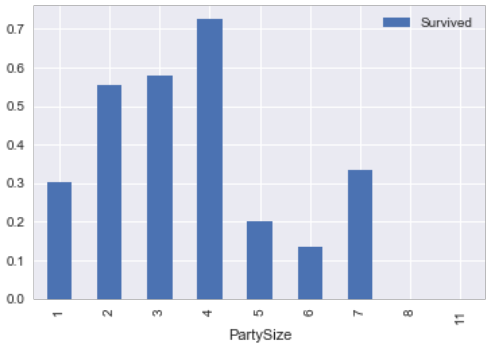
Now we can drop age and use age range instead: *[Pclass, Title, Sex, AgeRange, SibSp, Parch, Fare, Embarked]*

Step 7 SibSp&Parch



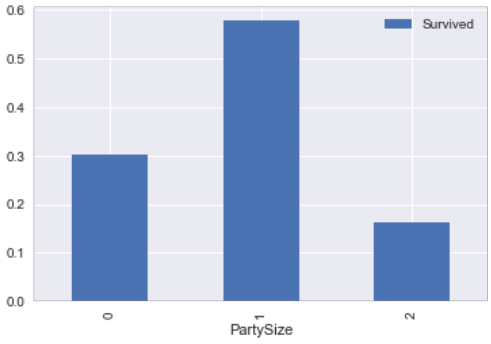


The relationships of SibSp and Parch with survival are similar, we can combine then to for "PartySize"



Further grouping certains sizes we can form three ranges for party size:

Alone: 1
Small: 2, 3, 4
Large: >= 5



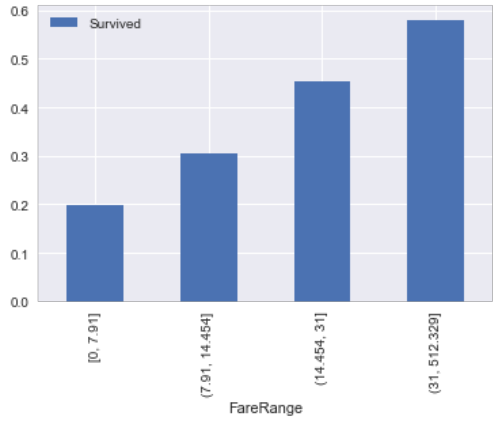
Now we see it clear that small size parties have better chance of survival than big size parties. We can drop SibSp and Parch and have PartySize instead.

[Pclass, Title, Sex, AgeRange, PartySize, Fare, Embarked]

Step 8 Fare

Fare is similar to Age, which has big relationship with the survival chance but spreads out. So I splited the fare into four ranges:

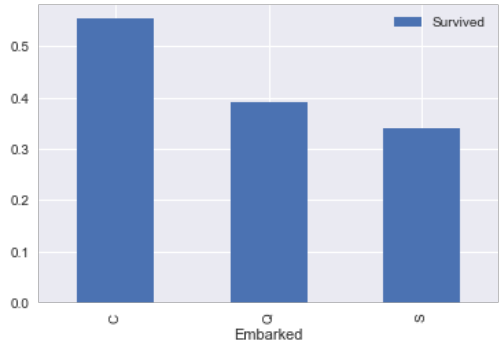
[0-7.91], [7.91-14.525], [14.525, 31], [31, 522]



We can drop Fare and use FareRange instead.

[Pclass, Title, Sex, AgeRange, PartySize, FareRange, Embarked]

Step 9 Embarked



We see a correlation of embankment with the survival, we can keep this for feature use.

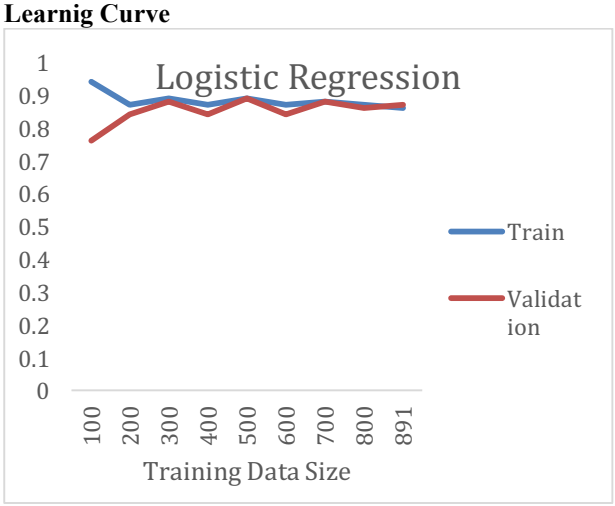
Finally we can determine based on the following features:

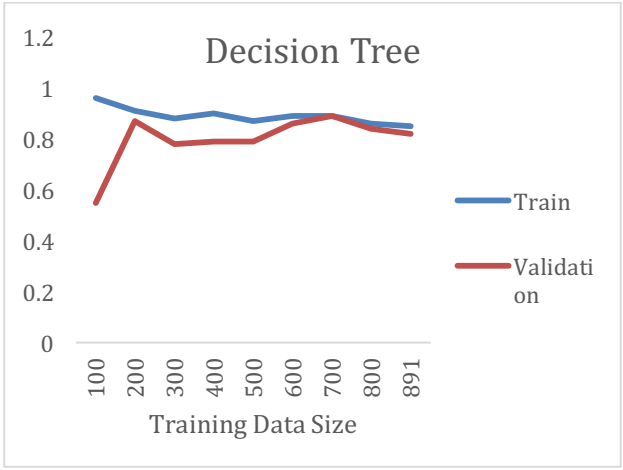
[Pclass, Title, Sex, AgeRange, PartySize, FareRange, Embarked]

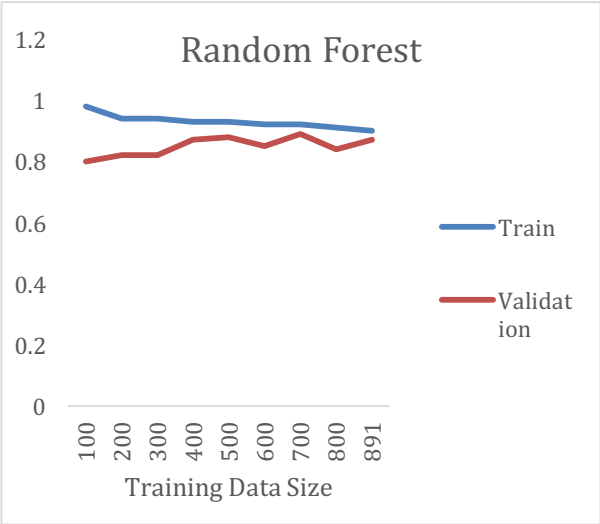
b) *Build at least two of {Naïve Bayes, Logistic Regression, random forests, support vector machines or neural networks using the libraries of Spark.MLLib only. Explain your choice; plot learning curves; explain observed behavior; investigate which features are most informative; do at least one round of error analysis to maximize your chosen metric (F1, accuracy, weighted F1); explain your choice of metric.*

Model	Accuracy
Logistic Regression	0.878809869376
Decision Tree	0.868015239478
Random Forest	0.883708272859

Learnig Curve







Error Analysis

1. Insert mean/mode for missing values

	Before	After
Logistic Regression	0.82	**0.84**
Decision Tree	0.75	**0.79**
Random Forest	0.83	**0.84**

2. Use PartySize

Before I was ignoring the sibsp and parch. Then I discovered that adding them up we can get the party size. With consideration of the party size and survival chance. We are able to achieve a higher accuracy:

	Before	After
Logistic Regression	0.82	**0.88**
Decision Tree	0.79	**0.87**
Random Forest	0.84	**0.88**

c) *Complete an analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.*

As analyzed before, we can predict whether a passenger survived based on his/her features:

[Pclass, Title, Sex, AgeRange, PartySize, FareRange, Embarked]

Please see submission.csv for the prediction for test. We can extract a list of passengers on prediction = 1.

III. Tier 1 Addendum

This part is using Microsoft Excel and code in ***FeatureEngineering.ipynb***

a) *Statistical Summary*

There are two files: **train.csv** and **test.csv**. Here is brief summary of the files:

DataType	Notes
Survival	Survived or not
Pclass	Passenger class, a proxy for socio-economic status
Name	
Sex	
Age	
SibSp	Number of Siblings/Spouses Aboard. Sibling: Brother, Sister, Stepbrother, or Stepsister of Passenger Aboard Titanic. Spouse: Husband or Wife of Passenger Aboard Titanic (Mistresses and Fiances Ignored)
Parch	Number of Parents/Children Aboard. Parent: Mother or Father of Passenger Aboard Titanic. Child: Son, Daughter, Stepson, or Stepdaughter of Passenger Aboard Titanic.
Fare	Passenger Fare
Cabin	Passenger Cabin Number
Ticket	Ticket Number
Embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

	Train.csv	Test.csv
Entries	891	419
Survived	549 – 61.6%	N/A
Pclass	1:216, 2:184, 3:491	1:107, 2:93, 3:218
Sex	M:314, F:577	M:266, F:152
Age	Range[0.75, 63] 177 missing	Range[0.17, 76] 86 missing
SibSp	0, 1, 2, 3, 4, 5, 8	0, 1, 2, 8
Parch	0-6	0, 1, 2, 4, 9
Ticket	Varies	Varies
Cabin	Varies, largely missing	Varies, largely missing
Fare	Varies, ranging from 0 – 512.3292	Varies ranging from 0 to 69.55
Embarked	C:168, Q:77, S:644, Blacks:2	C:102, Q:46, S:270

Chart 1

b) *Computation Requirements*

The computation requirement for this assignment is very low. Since we have less than 1000 entries in the training data and less than 500 in the test, and

the feature dimension is less than 10. Computing the accuracy for the entire. Here is the runtime for data processing and learning. Note that the learning part consists of fitting the model on the entire training set and then predicting the results:

2015 Model MacBook Pro	Time (s)
Logistic Regression	0.776620864868
Decision Tree	1.55955696106
Random Forest	0.887764930725
Feature Engineering	18.1202449799

From the above form we can see that the computation requirement is very low. This assignment can be easily handled by a personal laptop.

IV. Tier 2 Addendum

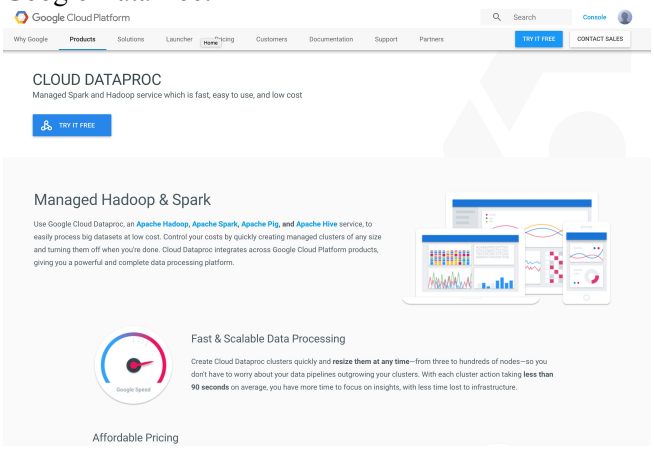
a) Visual evidence that the student used Spark running on Amazon Web Services (AWS) to complete the project, a summary stating the options for running Spark on AWS versus other cloud environments.

Below are the screenshots of my spark job running on AWS:

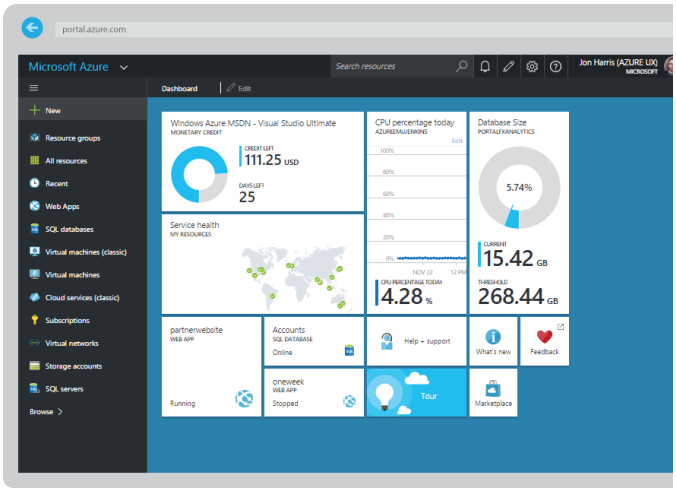
Please see attached:
console.jpg
aws.log
dashboard.jpg

Besides the AWS environment, there is also the Google Cloud and Microsoft Azure.

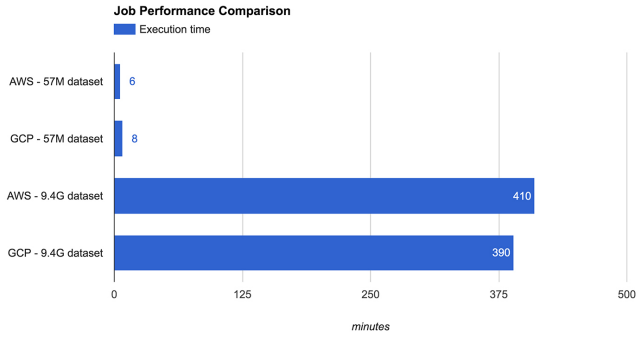
Google DataProc:

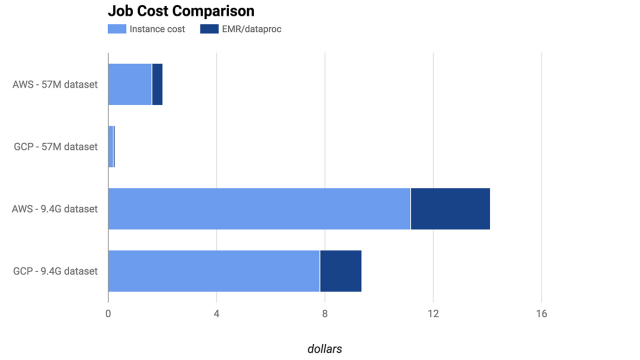


Microsoft Azure:



Here I found an interesting comparison between AWS EMR and Google Cloud: https://www.oreilly.com/ideas/spark-comparison-aws-vs-gcp





From the charts we can see that the performance per dollar of Google is higher than AWS. This is interesting finding and worth looking into. Over all the three major cloud provider are all good choices for Spark Hadoop jobs. One can choose their favorite based on the price, performance, and user experience.

b) *Discussion of the desirability for use of machine learning model types available in Spark.MLlib to complete the project.*

Here are my imports:

```
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer
from pyspark.ml.linalg import DenseVector
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.classification import DecisionTreeClassifier, RandomForestClassifier
```

My usage of StringIndexer to convert String features to numeric features:

```
# Index Features
cats = ["Pclass", "Title", "Sex", "Embarked"]
def indexer(data, col):
    res = StringIndexer(inputCol = col, outputCol = col + "I").fit(data)
    return res

indexers = [indexer(combined, col) for col in cats]
pipeline = Pipeline(stages = indexers)
combined = pipeline.fit(combined).transform(combined)
for i in ["Pclass", "Title", "Sex", "Embarked", "Fare", "SibSp", "Parch", "Age", "Name"]:
    combined = combined.drop(i)

print(combined.show(5))
```

```
+--------+--------+--------+---------+---------+-------+------+----+---------+
|Survived|Datatype|AgeRange|PartySize|FareRange|PclassI|TitleI|SexI|EmbarkedI|
+--------+--------+--------+---------+---------+-------+------+----+---------+
|     0.0|   train|       1|        1|        0|    0.0|   0.0| 0.0|      0.0|
|     1.0|   train|       2|        1|        3|    1.0|   2.0| 1.0|      1.0|
|     1.0|   train|       1|        0|        1|    0.0|   1.0| 1.0|      0.0|
|     1.0|   train|       1|        1|        3|    1.0|   2.0| 1.0|      0.0|
|     0.0|   train|       1|        0|        1|    0.0|   0.0| 0.0|      0.0|
+--------+--------+--------+---------+---------+-------+------+----+---------+
```

My usage of DenseVector to convert data into label/features vector:

```
# Convert To Label/Features Vector
nums = ["Survived", "AgeRange", "PartySize", "FareRange"]
catsI = [i + "I" for i in cats]
featCol = nums + catsI
featCol.remove("Survived")
labelCol = ["Datatype", "Survived"]
row = Row("datatype", "label", "features")

combined = combined[labelCol + featCol]
lf = (combined.rdd.map(lambda r : (row(r[0], r[1], DenseVector(r[2:])))).toDF())
lf = (StringIndexer(inputCol = "label", outputCol = "index").fit(lf).transform(lf))
lf.show(5)
```

```
+--------+-----+--------------------+-----+
|datatype|label|            features|index|
+--------+-----+--------------------+-----+
|   train|  0.0|[1.0,1.0,0.0,0.0,...|  0.0|
|   train|  1.0|[2.0,1.0,3.0,1.0,...|  1.0|
|   train|  1.0|[1.0,0.0,1.0,0.0,...|  1.0|
|   train|  1.0|[1.0,1.0,3.0,1.0,...|  1.0|
|   train|  0.0|[1.0,0.0,1.0,0.0,...|  0.0|
+--------+-----+--------------------+-----+
only showing top 5 rows
```

Here is my usage of machine learning models:

```
# Logistic Regression
start = time.time()
log = LogisticRegression(labelCol="index").fit(train)
pred = log.transform(validate)
evl = BinaryClassificationEvaluator(labelCol="index")
delta = time.time() - start
print("Logistic Regression " + str(evl.evaluate(pred)))
print("Time: " + str(delta))
```

```
Logistic Regression 0.863908368396
Time: 1.18547797203
```

```
# Decision Tree
start = time.time()
dtree = DecisionTreeClassifier(labelCol="index").fit(train)
pred = dtree.transform(validate)
evl = BinaryClassificationEvaluator(labelCol="index")
delta = time.time() - start
print("Decision Tree " + str(evl.evaluate(pred)) + " Time: " + str(delta))

# Random Forest
start = time.time()
randForest = RandomForestClassifier(labelCol="index").fit(train)
pred = randForest.transform(validate)
evl = BinaryClassificationEvaluator(labelCol="index")
delta = time.time() - start
print("Random Forest " + str(evl.evaluate(pred)) + " Time: " + str(delta))
```

```
Decision Tree 0.665451145395 Time: 1.05240416527
Random Forest 0.861360448808 Time: 1.32591080666
```

I have used three models from the Spark.ml: Logistic Regression, Decision Tree and Random Forest. Overall I feel the Spark.ml is very powerful.

c) *A maximum 1-page addendum to the report that analyzes the value of each "column" or feature of the data available in the Titanic data set.*

Raw:

```
root
 |-- PassengerId: string (nullable = t
rue)
 |-- Survived: double (nullable = true)
 |-- Pclass: string (nullable = true)
 |-- Name: string (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- SibSp: double (nullable = true)
 |-- Parch: double (nullable = true)
 |-- Ticket: string (nullable = true)
 |-- Fare: double (nullable = true)
 |-- Cabin: string (nullable = true)
 |-- Embarked: string (nullable = true)
 |-- Datatype: string (nullable = fals
e)
```

More statistical analysis can be found in part III.

After processing, here are the features:
More details on how these are derived can be found in part III

AgeRange	Range of the passenger's age: 0: < 15 1: 15 - 35 2: > 35
PartySize	Number of relatives in

	travelling together: 0: travelling alone 1: with 1-3 people, medium party 2: with 4 or more people, large party
FareRange	Range of the passenger fare: 0: < 7.91 1: 7.91 – 14.54 2: 14.54 - 31 3: > 31
Embarked	Port of Embarkation: 0: C 1: Q 2: S
Sex	0: female 1: male
Title	Title in name: 0: Mr 1: Miss 2: Mrs 3: Master 4: Others
Pclass	Passenger class: 0: 1 1: 2 2: 3

V. TIER 3 REFLECTIONS

a) *A maximum 1-page discussion of the value of this assignment. Why would we assign it? What value should you obtain from it? How does this exercise compare with selecting data science tools for a first project at a recently formed Startup company?*

Why do we assign it?

In my opinion, this assignment provides a very good hands-on experience with a basic data science project. Through this assignment I can put my skills into practice by solving real world problems and learn to use the Spark frame work as well as AWS platform.

What value should you obtain from it?

Throughout this assignment I learned the skill to analysis and process data that came in raw and dirty stage. I can use tools like Spark and AWS to help process data with cloud computing. Also I get to try models from Spark.ml library on real data, plot learning curves and analyze errors. Overall I got a very good hands-on experience with the Spark pipeline to solve a real word problem.

How does this exercise compare with selecting data science tools for a first project at a recently formed Startup company?

In this assignment, I used the current tools and framework (Spark and AWS) to solve the real word data problem. The tools are going to benefit me in doing all other data related problem solving. Also the experience gained in processing and analyzing the Titanic dataset can help me in other data processing and analyzing works in the future. The work load is equivalent to what to expect in a startup company project. Moreover, the experience of using the pipeline to solve the problem end to end is very important for being in a self-driven environment like a startup.

b) *A maximum 1-page discussion with suggestions for how we could improve the assignment.*

One improvement I can think of is to have a clearer guideline/requirement. In the description I am seeing several questions with similar or ambiguous ask. I understand that in companies the objectives can be general. However, as a student I hope to know exactly what the assignment requiems are.

c) *A maximum 2-page discussion describing how you will build a data science pipeline for analyzing large and small data sets for this course.*

For small data sized project:

1. *Data processing*
   Look at the raw data, try to understand the entries. Use statistical tools (excel) to "play with" the data and get correlations with the entries. This can be done on my local machine.
2. *Feature Engineering*
   Take the entire data set, find correlations of the entries and labels. Plot graphs if necessary as done in this assignment. Reduce entries to features convert string to numeric, define good rages and sub categories for some entries. This can be done on local machines.
3. *Build Models*
   After features are determined, build models using necessary libraries. Find models that fits the feature and data sets. This can be done locally if data is small.
4. *Cross Validation*

Do cross validations, plot learning curves and revisit the models.

5. *Error Analysis*
   For errors, analysis based on a subset of them or on the cloud. Find the ways to improve and then come back to the models. Refine the feature and models if necessary.

For large data sized projects:

1. *Data processing*
   Look at a subset of raw data, try to understand the entries. Use statistical tools (excel) to "play with" the data and get correlations with the entries. I can do this on my local machine. Also, I can read the description of the data files if any. Last, I can do statistical analysis on AWS machines of the entire data.
2. *Feature Engineering*
   Take a subset of the data set, find correlations of the entries and labels. Plot graphs if necessary as done in this assignment. Reduce entries to features convert string to numeric, define good rages and sub categories for some entries. Once the features are determined, I can use Spark and AWS to process the entire data set features.
3. *Build Models*
   After features are determined, build models using necessary libraries. Find models that fits the feature and data sets. This should be done using Spark.ml
4. *Cross Validation*
   Do cross validations, plot learning curves and revisit the models.
5. *Error Analysis*
   For errors, analysis based on a subset of them or on the cloud. Find the ways to improve and then come back to the models. Refine the feature and models if necessary.