Flight Data Preview

1. Overall introduction of the project

The flight data set used in the experiments in this section is the flight punctuality statistical data provided on the 2009 Data Expo. Through exploring the data, we know that the data has a total of 29 fields. Based on information such as departure times, departure/arrival delays, etc., we have the following questions:

What are the busiest times of day for flights? There are usually extreme weather such as heavy fog in the morning and evening. Are there more arriving and departing flights at noon?

Which is the most punctual? When designing a travel plan, if there are two adjacent airports to a certain destination, we seem to be able to compare which is more punctual, so as to reduce the impact of possible delays on our travel.

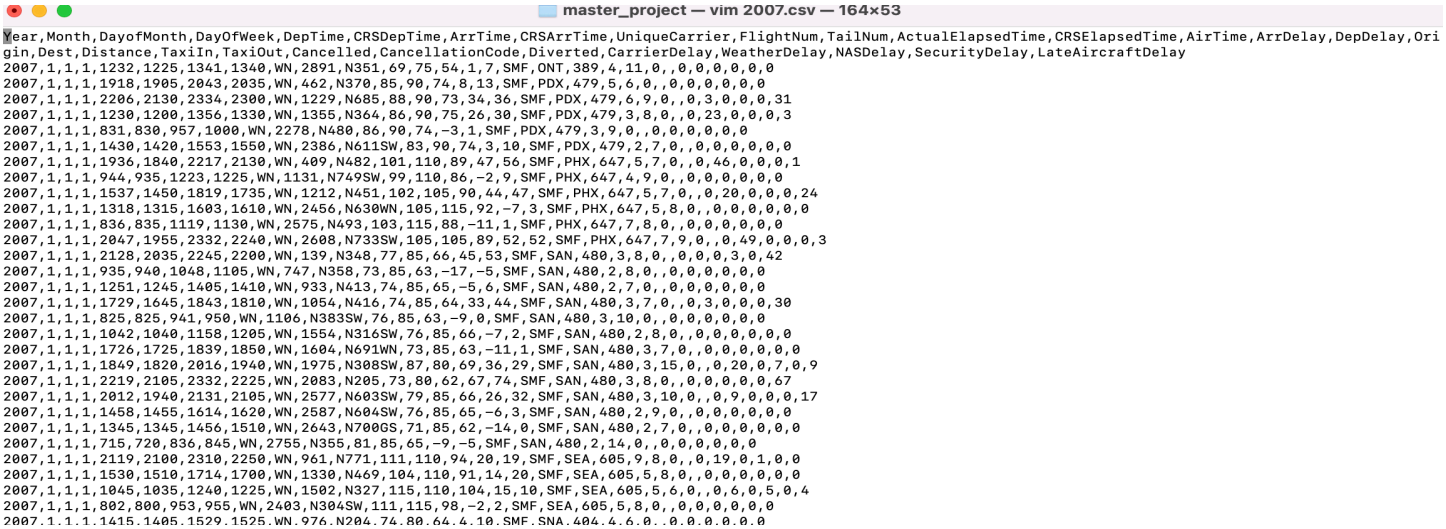
What are the hardest hit areas for departure delays?

Moreover, we will demonstrate how to build a predictive model using Spark and ML-Lib.

This project uses Apache Spark through its Scala API to generate our feature matrix and ML-Lib (Spark's machine learning library) to build and evaluate our classification model.

Building a predictive model of flight delays, the source dataset is at our download, which includes details on US flights between 2007 and 2008.

We will build a supervised learning model to predict delays of flights leaving O'Hare International Airport (ORD). Finally we will use the 2007 data to build the model and use the 2008 data to test its validity.



Data schema

n.	Forbidden	Name	Description
1		Year	1987-2008
2		Month	1-12
3		DayofMonth	1-31
4		DayOfWeek	1 (Monday) - 7 (Sunday)
5		DepTime	actual departure time (local, hhmm)
6		CRSDepTime	scheduled departure time (local, hhmm)
7	x	ArrTime	actual arrival time (local, hhmm)
8		CRSArrTime	scheduled arrival time (local, hhmm)
9		UniqueCarrier	unique carrier code
10		FlightNum	flight number
11		TailNum	plane tail number
12	x	ActualElapsedTime	in minutes
13		CRSElapsedTime	in minutes
14	x	AirTime	in minutes
15		ArrDelay	arrival delay, in minutes
16		DepDelay	departure delay, in minutes
17		Origin	origin IATA airport code
18		Dest	destination IATA airport code
19		Distance	in miles
20	x	TaxiIn	taxi in time, in minutes
21		TaxiOut	taxi out time in minutes
22		Cancelled	was the flight cancelled?
23		CancellationCode	reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24	x	Diverted	1 = yes, 0 = no
25	x	CarrierDelay	in minutes
26	x	WeatherDelay	in minutes
27	x	NASDelay	in minutes
28	x	SecurityDelay	in minutes
29	x	LateAircraftDelay	in minutes

Q1: Count the number of flights in different time range:

```
println("The number of flight departing in the early morning: ")
sqlContext.sql( sqlText = "SELECT COUNT(data.FlightNum) FROM data WHERE data.DepTime BETWEEN 0 AND 600").show()
println("The number of flight departing in the morning: ")
sqlContext.sql( sqlText = "SELECT COUNT(data.FlightNum) FROM data WHERE data.DepTime BETWEEN 601 AND 1000").show()
println("The number of flight departing in the noon: ")
sqlContext.sql( sqlText = "SELECT COUNT(data.FlightNum) FROM data WHERE data.DepTime BETWEEN 1001 AND 1400").show()
println("The number of flight departing in the afternoon: ")
sqlContext.sql( sqlText = "SELECT COUNT(data.FlightNum) FROM data WHERE data.DepTime BETWEEN 1401 AND 1900").show()
println("The number of flight departing in the evening: ")
sqlContext.sql( sqlText = "SELECT COUNT(data.FlightNum) FROM data WHERE data.DepTime BETWEEN 1901 AND 2359").show()
```

Results:

```
The number of flight departing in the early morning:
+--------+--------+
|count(FlightNum)|
+----------------+
|          191867|
+----------------+

The number of flight departing in the morning:
+----------------+
|count(FlightNum)|
+----------------+
|         1866083|
+----------------+

The number of flight departing in the noon:
+----------------+
|count(FlightNum)|
+----------------+
|         1827918|
+----------------+

The number of flight departing in the afternoon:
+----------------+
|count(FlightNum)|
+----------------+
|         2224672|
+----------------+

The number of flight departing in the evening:
+----------------+
|count(FlightNum)|
+----------------+
|         1181155|
+----------------+
```

Q2: Find the top 5 destinations has most flights flight to on time:

```
val queryDestResult = sqlContext.sql( sqlText = "SELECT DISTINCT data.Dest, COUNT(data.ArrDelay) " +
  "AS delayTimes FROM data where data.ArrDelay = 0 GROUP BY data.Dest ORDER BY delayTimes DESC")
println(queryDestResult.head(5).mkString("Array(", ", ", ")"))
```

Results:

```
22/12/18 22:05:50 WARN package: Truncated the string representatio
Array([ATL,10534], [DFW,7692], [ORD,7383], [PHX,7089], [DEN,6715])
```

Q3: Find the top 5 departure place has most flights flight from delayed:

Results:

```
22/12/18 22:21:31 INFO BlockManager: Initialized BlockManager: Bl
22/12/18 22:21:31 INFO ContextHandler: Started o.s.j.s.ServletCon
StructType(StructField(Year,StringType,true), StructField(Month,S
22/12/18 22:21:36 WARN package: Truncated the string representati
Array([RNO,999], [PHL,999], [EWR,999], [RST,998], [SWF,997])
```

Q4: Flight Delay Predictor

Q4: Flight Delay Predictor

Data Preprocess

- We will perform the same preprocessing using a Spark RDD to convert the raw flight delay dataset into two feature matrices: data_2007 (our training set) and data_2008 (our test set).
- Using DelayRec, our processing will perform the following steps (in function prepFlightDelays):

1. We use Spark's SparkContext.textFile method to read the original input file to generate an RDD.
2. Each row is parsed into fields using CSVReader and filled into the DelayRec object
3. We then perform a series of RDD transformations on the input RDD to ensure we only have rows corresponding to flights that were not canceled and originated from the ORD.
4. Finally, we use the gen_features method to generate the final feature vectors for each row as a set of doubles.

Use Spark and ML-lib to build models

Use data_2007 dataset (for training) and data_2008 dataset (for validation) as RDD, and then use Spark's ML-Lib machine learning library to build a predictive model.

ML-Lib is Spark's extensible machine learning library that includes various learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, and more.

To use ML-Lib's machine learning algorithms, first we parse our feature matrix into an RDD of LabeledPoint objects (for the training and test datasets). LabeledPoint is ML-Lib's abstraction of labeled feature vectors.

We consider flight delays of 15 minutes or more as "delayed" and mark them as 1.0, and those within 15 minutes as "non-delayed" and mark them as 0.0.

We also use ML-Lib's StandardScaler class to normalize the feature values of the training and validation sets. This is important because ML-Lib uses stochastic gradient descent, which performs best if the feature vectors are normalized.

Results:

```
(1.0,[-1.6160463330366632,1.3961052168540335,1.5354307758475594,0.362432098412101,0.4316551188434339,-0.0232
(1.0,[-1.6160463330366632,1.5098311893165113,-1.471090248772824,1.4641277433573872,-0.7436888225169946,0.062
22/12/18 22:41:21 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
22/12/18 22:41:21 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
Logistic Regression - precision = 0.55, recall = 0.06, F1 = 0.11, accuracy = 0.65
SVM model precision = 0.44, recall = 0.69, F1 = 0.54, accuracy = 0.58
Decision Tree precision = 0.45, recall = 0.32, F1 = 0.37, accuracy = 0.62
Random forest precision = 0.48, recall = 0.20, F1 = 0.28, accuracy = 0.64
```