# Sage Plagarism Report

Powered by schoolworkspro.com

Krishna Prasad Adhikari
**150265**
**Word Count: 5680**

33
Results Found

12.53 %
Match Percentage

| | |
|---|---|
| Submitted to: Softwarica | **6.31%** |
| Submitted to: Softwarica | **0.48%** |
| Source: What is Exploratory Data Analysis GeeksforGeeks Link: https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/ | **0.32%** |
| Submitted to: Softwarica | **0.28%** |
| Source: What is Data Analytics and its Future Scope in 2022 Link: https://www.simplilearn.com/tutorials/data-analytics-tutorial/what-is-data-analytics | **0.26%** |
| Submitted to: Softwarica | **0.26%** |
| Source: Spark SQL and DataFrames Spark 220 Documentation Link: https://spark.apache.org/docs/2.2.0/sql-programming-guide.html | **0.26%** |
| Submitted to: Softwarica | **0.25%** |
| Submitted to: Softwarica | **0.25%** |
| Submitted to: Softwarica | **0.25%** |
| Submitted to: Softwarica | **0.25%** |
| Submitted to: Softwarica | **0.23%** |

| | |
|---|---|
| Submitted to: Softwarica | **0.23%** |
| Submitted to: Softwarica | **0.23%** |
| Source: What is Machine Learning? \| Types of Machine Learning \| Edureka<br>Link: https://www.edureka.co/blog/what-is-machine-learning/ | **0.23%** |
| Submitted to: Softwarica | **0.21%** |
| Submitted to: Softwarica | **0.21%** |
| Submitted to: Softwarica | **0.19%** |
| Source: Drawing on research and data<br>Link: https://en.wikipedia.org/wiki/Descriptive_statistics | **0.18%** |
| Submitted to: Softwarica | **0.18%** |
| Submitted to: Softwarica | **0.18%** |
| Submitted to: Softwarica | **0.16%** |
| Submitted to: Softwarica | **0.16%** |
| Submitted to: Sunway | **0.14%** |
| Submitted to: Softwarica | **0.14%** |
| Source: CV preparation assignment<br>Link: https://en.wikipedia.org/wiki/John_F._Kennedy | **0.12%** |
| Submitted to: Softwarica | **0.12%** |
| Submitted to: Sunway | **0.11%** |
| Submitted to: Softwarica | **0.11%** |
| Submitted to: Softwarica | **0.09%** |
| Source: Biometric Considerations<br>Link: https://en.wikipedia.org/wiki/Border_control | **0.07%** |

Submitted to: Softwarica **0.05%**

Submitted to: Softwarica **0.02%**

Table of Contents

Abstract..... 1
Introduction ..... 1
PySpark ..... 2
Spark Installation ..... 2
Tableau ..... 5
Dataset Analysis ..... 6
Exploratory Data Analysis (EDA) ..... 7
Data Pre-processing ..... 7
Data Analysis ..... 8
Data Visualization ..... 11
Flight Delay Prediction using Spark ML .....15
Training Dataset .....15
Testing Dataset .....16
Logistics Regression .....17
Support Vector Machine (SVM) .....17
Data Visualization Using Tableau .....19
Conclusion .....22
References .....23
Appendix .....24

Air transportation is more popular than road transportation for longer distance as it is considered fastest
mode of transportation.
Air transportation is also considered the safest mode of transportation
(Lawyers, 2022).
With the increase in numbers of passengers, number of airlines increases, and number
of flights increases with increase in airlines.
The increasing number of airlines makes them challenging in
maintaining the flow of passengers.
To attract passenger airlines are forced to improve the quality of
service, better reliability in terms of safety and time.
(Lawyers, 2022)
In this paper, we will analyze the flights and airports from different possible aspect. Flight delay is
considered as the center of data analysis using PySpark.
PySpark is also used to implement the
techniques of machine learning and prediction.
We are using open dataset of flights and detail data
related to flights and airports.
Commercial airlines are generally considered one of the safest and fastest modes of public
However, flight delay is experienced frequent all around the globe due to various factor
like Weather, Air Traffic Congestion, Technical Issues, Security checks, Baggage handling and more.
project, dataset of airports and flight detail with delay in arrival and departure is considered with the aim
to perform machine learning with data frame and spark SQL.
The result from the project might be useful
for the analysis of the airlines performance and result of which might be helpful for the airlines to
improve in services.
(turbi, 2022)
In this paper detail explanation and result demonstration of the project is explaining. Firstly, detail
explanation of programs and environments setup are discussed which includes step by step installation
process and its purpose.
Secondly, the data frame is explained along with the source of data collection.
After that the crucial codes and result obtained are attached with explanation.
All codes are not included
in the description as similar codes are executed multiples times.
However, the complete source code is
attached in the appendix section.

The selected dataset is preprocessed as a data cleaning, reduction, and transformation.

Explorative Data
Analysis (EDA) is performed on the preprocessed dataset and visualization of the findings are also
performed using PySpark.
For the visualization, both spark and tableau are used After Preprocessing,
EDA and Visualization the eventual goal of the project is the use of machine learning algorithm for the
The results are finally further analyzed and discussed along with the possible areas of further
Pyspark is an interface of Apache Spark in Python.
It allows to write Spark applications using Python but
also provides PySpark shell for analysis of the data in the distributed environment.
PySpark also supports

Spark's features such as Spark SQL, DataFrame, Streaming, Machine Leaning and Spark Core.

Figure 1: Apache Spark Architecture

With PySpark, easily integration and work with Resilient Distributed Dataset (RDD) in Python Programing
Language can be done which make PySpark an efficient framework for working with massive datasets.
can be used to perform computations on large datasets or just analyze them.
Real time computations,
Polyglot caching and disk persistence, fast processing, and compatibility with RDDs are the major key
features of PySpark.
In this project, PySpark is used for Data Cleaning, EDA and also Machine learning and data visualization.
(intellipaat, 2023)
Spark Installation
Installation of Spark in possible is Windows, Linux and MAC OS.
However, for this project, Spark is
installed in physical windows machine.
Detail steps of installation is illustrated and described in this
Step 1.
Java Installation
Apache Spark requires Java 8 or newer version is required.
64bit version of Java 8 was downloaded and
installed from the official webpage of oracle.

(https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html)

Installed Java version can be checked and confirmed from command prompt
Figure 2.
Java version check from command prompt

Step 2.
Python Installation
Python was installed from python official website(https://www.python.org/downloads/release/python-
After completion of python installation, installed python can be confirmed from the command
Figure 3.
Python version check from command prompt
Step 3.
Apache Spark Installation
Apache Spark was downloaded from the official website and extracted to C:\apps\opt\spark-3.4.1-bin-
Step 4.
Configure Environment Variables
After installing Java, Python and Apache Spark Environment variables JAVA_HOME, SPARK_HOME,
HADOOP_HOME and PATH were set.
Figure 4.
Environment Configuration
Step 5.
Execute Pyspark
After installation of Spark, command “pyspark” was executed in cmd in C:\apps\opt\spark-3.4.1-bin-
hadoop3\bin
Figure 5.
Start Pyspark
Step 6.
Installation of Jupiter Notebook
To perform PySpark operation, Anaconda Navigator was installed where Jupyter Notebook extensions
Figure 6.
Jupyter Notebook in Anaconda Navigator
All packages required to execute the project was installed and executed.
Tableau is a data visualization too used to data analysis and business intelligence.
It makes the analysis of
large set of data easier and faster.
Graphical plot of data such as charts, graphs can be plot with help of
tableau which simplify the visualization and analysis of data easier and helps in making business
In this project, tableau is also used for the data visualization and analysis.
Dataset Analysis
Dataset “flights.csv” was selected from Kaggle
(https://www.kaggle.com/datasets/matinsajadi/flights)
which is publicly available and contains information about American airlines and airports of the year

This dataset contains information about flight company, flight number, origin and destination,
flight duration, distance, arrival time, delay with exact data and time of the flight.
There are 21 column
and 336775 rows data.
Feature Description Datatype
id A unique identifier for each flight record Integer
year Year in which the flight took place Integer
month Month in which the flight took place Integer
day Day in which the flight took place Integer
dep_time Departure Time of the flight in 24-hour format (hhmm) Float
sched_dep_time Scheduled local departure time in 24-hour format (hhmm) Integer
dep_delay Departure delay time where positive is delayed and negative is early departure
arr_time Arival time of light in 24-hour format (hhmm) float
sched_arr_time Scheduled local arrival time in 24-hour format (hhmm) Integer
arr_delay Arrival delay time where positive is delayed and negative is early arrival
carrier Code of airlines carrier String
flight Flight number of the flight Integer
tailnum Unique identifier of the aircraft used for the flight String
origin Origin Airport Code String
dest Destination Airport Code String
air_time Flight time in year in minute Float
distance Distance between the origin and destination airports in miles Integer
hour Hour component of the scheduled departure time, in local time Integer
minute Minute component of the scheduled departure time, in local time Integer
time_hour Scheduled departure time of the flight, in local time (yyyy-mm-dd hh:mm:ss)
name Name of airlines carrier String
The available dataset contains different datatypes including integer, float, string and DateTime.
the dataset null or empty data are also expected.
Also, datatype and entered data in cell might have
been mismatched which need to be processed and remove the mismatched and unwanted data by the
process called preprocessing.
Pre-processed data can be used to analyze airlines, service quality, flight
Exploratory Data Analysis (EDA)
Exploratory Data Analysis (EDA) is an approach used to analyze the data and track down the trend and
patterns from the datasets.
Also, EDA helps in finding statistical summaries and graphical visualizations.
Data Pre-processing
The collected data sample was considered from publicly available dataset, the dataset is expected with

inconsistence data.
To Perform EDA, the dataset needs to be preprocessed to obtain more relevant data analysis and visualization.
Initially PySpark was installed using pip.
After the installation of pyspark, spark session was created to perform task and data frame reader was
used to read the CSV file.
Resilient Distributed Dataset (RDD) is a basic data structure of Apache spark.
Each and every dataset in
RDD is logically partitioned across different servers.
Using this partitions, different nodes and cluster can
computer at the same time.
The flight dataset is repartitioned into 8 partitions.
As a part of preprocessing, null values from DataFrame are removed.
Null values in DataFrame might
mislead the information during data analysis.
Also duplicate rows needs to be removed to avoid
confusions, avoid unnecessary storage space and also calculation will be fast.
Na data from dataset was removed and no duplicate data was observed.
Data Analysis
After Preprocessing, data analysis was performed.
Data analysis is focused on different aspect of flight
For the analysis of data, DataFrames are persisted as table and Spark SQL was used to query them
using SQL language.
SQL Query was executed in cleaned dataset and the result was printed for analysis.
Firstly, the SQL query was run to get the arrival delay by Airlines in minutes.
attached below.
From the above obtained result, it is observed that average delay if Frontier Airlines has the highest
average delay whereas US Airways Inc has the lowest Average Delay whereas Hawaiian Airlines and
Alaska Airlines Inc. has the negative delay record which means that the airlines depart or arrived before
the estimated time.
Also, for further analysis, total flight count based on the origin and destination was extracted using Spark
SQL query.
From the obtained result, EWR has the highest number of departures followed by JFK and LGA.
query was executed for the destination airports.
From the result, it is observed that Airport ATL has the highest number of arrivals.
Since, Total departure and total arrivals was calculated, Delay in route of Departure and Arrival is

On analyzing the average delay of Route between source and destination, EWR to TYS has the highest
average delay of flights.
Further delay analysis was performed based on distance between Origin and destination.
The result obtained from average delay based on distance illustrates that longer the distance, lesser the
Data Visualization
Data analysis using Spark SQL Performed.
Data Regarding airlines, flights and delay was analyzed.
further illustration in the form of graphical view, data visualization was performed using pandas.
graphical representation helps to understand the data by summarizing and presenting the huge set of
data in a easy understood graph or plot.
Pandas, seaborn and matplotlib was imported for the graphical presentation.
To understand the airlines and flights count was illustrated.
The observation will help to analyze the size
of an airlines on the basis of number of flights.
Figure 7 Bar chart of number of flights by airlines
From the bar plot of Airlines and number of flights performed in the period of available dataset, it is
observed that UA (United Air Lines Inc.) has the highest number of flights in the given set time followed
by B6 (JetBlue Airways), EV (ExpressJet Airlines Inc.), DL (Delta Air Lines Inc.) and others.
Similarly, based on size of airlines, average delay of the airlines was visualized, the result obtained
illustrated the F9 (Frontier Airlines Inc.) which has very less flights compared to other airlines seems to
have the highest average delay.
Followed by FL (AirTran Airways Corporation) which is also smaller
airlines compared to other airlines.
Figure 8 Average flight delay in minuted by airlines
From the data available, total flights delay was calculated and illustrated in the pie chart.
observed that 78.53% of flights are not on schedule.
Similarly, 76.29% of the flight's arrival are not on
Figure 9 Departure and Arrival Delay in the dataset in percentage
For the further visualization of arrival and departure delay, the correlation between arrival delay,
departure delay and distance from origin to destination was plotted.
Figure 10 Correlation between arrival delay, departure delay and distance
From the correlation matrix, it is observed that the correlation between departure delay and arrival

delay is positively correlated with result of 91%.
Where the correlation between distance and arrival
delay was observed negative, which means that there is inverse relation between distance and arrival
Also, frequency of flights based on distance was visualized and the obtained result shows that short
length have the higher frequency as compared to long duration flights.
Figure 11 Frequency of flights by flight time duration
Also, on further visualizing the data based on distance and airtime, flights with longer duration and
longer distance have slightly less delay as compared to shorter duration and shorter distance flight.
Figure 12 Box plot to illustrate departure delay of flights by distance and air time
Flight Delay Prediction using Spark ML
Machine learning algorithms consists of algorithms that enables computer systems to make predictions
which helps to make decisions based on the prediction result obtained from applying machine learning
algorithms to the dataset.
For the dataset to feed into machine learning algorithm, the dataset goes
through multiple process of data collection, cleaning, analyzing, and interpreting the data to exact the
valuable information which will help in making decision.
(Pedro, 2023)
Training Dataset
For training the dataset, valuable information was selected.
Since, flight delay predictions is being
The data available in the dataset is in minutes, where delay time less than 15 minutes and
earlier departure are considered not delay.
Any flights with delay time longer than 15 minutes are
considered delay.
For building the supervised learning models, datasets are split into train and test mode.
70% of data is
selected for training and remaining 30% is reserved for testing.
Preparing Training data for Spark ML
A Predictive model required multiple staged of feature preparation.
A pipeline consists of multiple stages
for data transformation and estimation which will prepare data frame for modeling and then train the
predictive model.
In the project, pipeline is created with StringIndexer, VectorAssembler and MinMaxScaler.
categorical features are converted from string value by StringIndexer estimator.

Categorical features are
combined into single vector using VectorAssembler, MinMaxScaler normalizes the continuous numeric
features and LogisticsRegression classifier trains the classification model.
After preparation of training data, pipeline is run on training data to train the model.
Testing Dataset
Now, the model produced by the pipeline will apply all the stages in the pipeline for specific DataFrame
and apply the trained model for the delay prediction.
This method can be used to predict the delay
status of flights where labels are unknown, but in this project test data with known label value, so
prediction status can be compared to actual status.
Here, DataFrame is transformed using the pipeline for the prediction.
After Testing the dataset, classifier was evaluated computing the confusion matrices.
To access the performance of classification, area under ROC was reviewed.
The obtained result is 1 which means that the classifier can perfectly distinguish the positive and
negative class point.
Logistics Regression
Logistics Regression is a predictive analysis technique based on a probability idea of binary(Yes/No)
events occurring (Chandrasekaran, 2023).
In the Project Logistics regression is used for the prediction.
Form the result of logistics regression, the result of prediction is 99.5%.
Support Vector Machine (SVM)
Support Vector Machine (SVM) is an supervised learning algorithm used for classification and regression
It is used for classification problems in machine learning (Saini, 2021).
In this project SVM is
used for the prediction.
From the result of SVM, the result obtained is 85%.
Figure 13 Prediction result of different prediction model used
Comparing results of both Logistics Regression and SVM, Precision of SVM is slighter lower than Logistics
The reason of lower precision in SVM might be due to over fitting or under fitting when
training machine learning model.
Data Visualization Using Tableau
Dataset of flights are airport is visualized using tableau.
Initially, dataset is connected to tableau and different features from the dataset are visualized.
Figure 14 Flights Count based on carrier, Origin and destination airports

From the above chart, United Air Lines Inc. (UA) has the highest number of flights and Liberty
International Airport (EWR) is the busiest airport.
EWR has the highest number of flights to
O'Hare International Airport (ORD).
Similarly, EWR has the highest number of flights to LAX and
LGA has the highest number of flights to ATL.
Figure 15 Flights number by days
From the above visualized image, Saturday is the busiest day with 80.39% flight in a single day
Similarly, the trend of flight is reflected in the delay with 83.13% flights delay on
Further breaking down of flight delay based on airlines, it is observed that carrier EV (ExpressJet Airlines Inc.) has the highest delay count compared to other flights.
Now, visualizing the data of flights, arrival delay and departure delay.
It is seen that higher
number of flight delay is observed during June, July, and December.
The reason of flight delay is
due to summer vacation during June and July, and Christmas in December.
In conclusion, flight and airlines data is analyzed and visualized using PySpark and Spark SQL.
Machine Learning algorithm is used for the model prediction.
Linear Regression and SVM was
used for the prediction where result of prediction was 99.5%.
Tableau was used to explore the
dataset and flights; airlines and delay of flights was explored.

Chandrasekaran, M., 2023.
Logistic Regression for Machine Learning.
Available at: https://www.capitalone.com/tech/machine-learning/what-is-logistic-regression/
[Accessed 22 08 2023].
intellipaat, 2023.
Apache Spark with Python.
Available at: https://intellipaat.com/blog/tutorial/spark-tutorial/pyspark-tutorial/#:~:text=PySpark%20is%20considered%20an%20interface,in%20a%20distributed%20environment%20interactively.
[Accessed 29 08 2023].
Lawyers, T. I., 2022.
The Safest Forms Of Transportation.
Available at: https://www.tkinjurylawyers.com/safest-forms-of-transportation/
[Accessed 20 08 2023].
Parsian, M., 2022.
Data Algorithms with Spark.
First ed.
s.l.
:O'Reilly Media, Inc,.

Pedro, J., 2023.
First Steps in Machine Learning with Apache Spark.
Available at: https://towardsdatascience.com/first-steps-in-machine-learning-with-apache-spark-672fe31799a3#:~:text=Our%20objective%20is%20to%20learn,make%20each%20of%20these
[Accessed 20 08 2023].
Saini, A., 2021.
Guide on Support Vector Machine (SVM) Algorithm.
Available at: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/
[Accessed 23 08 2023].
turbi, 2022.
The Safest Transport Modes, Ranked by Statistics From 10 Years of Data.
Available at: https://turbli.com/blog/the-safest-transport-modes-ranked-by-statistics-from-10-years-of-data/
[Accessed 20 08 2023].

Complete Source Code and output of the project is attached in this appendix section.

- Data Import
- Data Preparation
- Data Analysis
- Training Model
- Testing Model
- Using machine learning Algorithm for Prediction

Data Import

```
#!pip install pyspark
# import the dataframe sql data types from pyspark.sql.types import * from pyspark.sql import SparkSession
# # flightSchema describes the structure of the data in the flights.csv file # flightSchema = StructType([ StructField("id", IntegerType(), False), StructField("year", StringType(), False), StructField("month", StringType(), False), StructField("day", StringType(), False), StructField("dep_time", StringType(), False), StructField("sched_dep_time", StringType(), False), StructField("dep_delay", StringType(), False), StructField("arr_time", StringType(), False), StructField("sched_arr_delay", IntegerType(), False), StructField("arr_delay", StringType(), False), StructField("carrier", StringType(), False), StructField("flight", IntegerType(), False), StructField("tailnum", StringType(), False), StructField("origin", StringType(), False), StructField("dest", StringType(), False), StructField("air_time", StringType(), False), StructField("distance", StringType(), False), StructField("hour", IntegerType(), False), StructField("minute", IntegerType(), False), StructField("time_hour", TimestampType(), False), StructField("name", StringType(), False), ])
```

```
# # spark session to perform all tasks # spark = SparkSession.builder.appName('150265_KrishnaPrasadAdhikari_STW7082CEM').getOrCreate()
# # Use the dataframe reader to read the file and # flights = spark.read.csv("flights.csv", schema=flightSchema, header=True, multiLine=True)
flights.show() +---+----+-----+---+--------+-------------+---------+--------+------------- -+---------+-------+------+-------+------+----+--------+--------+----+------ +-----------------+------------------+ |
id|year|month|day|dep_time|sched_dep_time|dep_delay|arr_time|sched_arr_delay|
arr_delay|carrier|flight|tailnum|origin|dest|air_time|distance|hour|minute| time_hour|
name| +---+----+-----+---+--------+-------------+---------+--------+------------- -+---------+-------+------+-------+------+----+--------+--------+----+------ +-----------------+------------------+ | 0|2013| 1| 1| 517.0|
515| 2.0| 830.0| 819| 11.0| UA| 1545| N14228| EWR| IAH| 227.0| 1400| 5| 15|2013-01-
01 05:00:00|United Air Lines ...| | 1|2013| 1| 1| 533.0| 529| 4.0| 850.0| 830| 20.0| UA|
1714| N24211| LGA| IAH| 227.0| 1416| 5| 29|2013-01-01 05:00:00|United Air Lines ...|
| 2|2013| 1| 1| 542.0| 540| 2.0| 923.0| 850| 33.0| AA| 1141| N619AA| JFK| MIA| 160.0|
1089| 5| 40|2013-01-01 05:00:00|American Airlines...| | 3|2013| 1| 1| 544.0| 545| -1.0|
1004.0| 1022| -18.0| B6| 725| N804JB| JFK| BQN| 183.0| 1576| 5| 45|2013-01-01
05:00:00| JetBlue Airways|
| 4|2013| 1| 1| 554.0| 600| -6.0| 812.0| 837| -25.0| DL| 461| N668DN| LGA| ATL| 116.0|
762| 6| 0|2013-01-01 06:00:00|Delta Air Lines Inc.| | 5|2013| 1| 1| 554.0| 558| -4.0|
740.0| 728| 12.0| UA| 1696| N39463| EWR| ORD| 150.0| 719| 5| 58|2013-01-01
05:00:00|United Air Lines ...| | 6|2013| 1| 1| 555.0| 600| -5.0| 913.0| 854| 19.0| B6| 507|
N516JB| EWR| FLL| 158.0| 1065| 6| 0|2013-01-01 06:00:00| JetBlue Airways| | 7|2013|
1| 1| 557.0| 600| -3.0| 709.0| 723| -14.0| EV| 5708| N829AS| LGA| IAD| 53.0| 229| 6|
0|2013-01-01 06:00:00|ExpressJet Airlin...| | 8|2013| 1| 1| 557.0| 600| -3.0| 838.0| 846|
-8.0| B6| 79| N593JB| JFK| MCO| 140.0| 944| 6| 0|2013-01-01 06:00:00| JetBlue
Airways| | 9|2013| 1| 1| 558.0| 600| -2.0| 753.0| 745| 8.0| AA| 301| N3ALAA| LGA| ORD|
138.0| 733| 6| 0|2013-01-01 06:00:00|American Airlines...| | 10|2013| 1| 1| 558.0| 600|
-2.0| 849.0| 851| -2.0| B6| 49| N793JB| JFK| PBI| 149.0| 1028| 6| 0|2013-01-01
06:00:00| JetBlue Airways| | 11|2013| 1| 1| 558.0| 600| -2.0| 853.0| 856| -3.0| B6| 71|
N657JB| JFK| TPA| 158.0| 1005| 6| 0|2013-01-01 06:00:00| JetBlue Airways| |
12|2013| 1| 1| 558.0| 600| -2.0| 924.0| 917| 7.0| UA| 194| N29129| JFK| LAX| 345.0|
2475| 6| 0|2013-01-01 06:00:00|United Air Lines ...| | 13|2013| 1| 1| 558.0| 600| -2.0|
923.0| 937| -14.0| UA| 1124| N53441| EWR| SFO| 361.0| 2565| 6| 0|2013-01-01
06:00:00|United Air Lines ...| | 14|2013| 1| 1| 559.0| 600| -1.0| 941.0| 910| 31.0| AA|
707| N3DUAA| LGA| DFW| 257.0| 1389| 6| 0|2013-01-01 06:00:00|American Airlines...|
| 15|2013| 1| 1| 559.0| 559| 0.0| 702.0| 706| -4.0| B6| 1806| N708JB| JFK| BOS| 44.0|
187| 5| 59|2013-01-01 05:00:00| JetBlue Airways| | 16|2013| 1| 1| 559.0| 600| -1.0|
854.0| 902| -8.0| UA| 1187| N76515| EWR| LAS| 337.0| 2227| 6| 0|2013-01-01
06:00:00|United Air Lines ...| | 17|2013| 1| 1| 600.0| 600| 0.0| 851.0| 858| -7.0| B6| 371|
N595JB| LGA| FLL| 152.0| 1076| 6| 0|2013-01-01 06:00:00| JetBlue Airways| |
18|2013| 1| 1| 600.0| 600| 0.0| 837.0| 825| 12.0| MQ| 4650| N542MQ| LGA| ATL| 134.0|
762| 6| 0|2013-01-01 06:00:00| Envoy Air|
| 19|2013| 1| 1| 601.0| 600| 1.0| 844.0| 850| -6.0| B6| 343| N644JB| EWR| PBI| 147.0|
1023| 6| 0|2013-01-01 06:00:00| JetBlue Airways| +---+----+-----+---+--------+-------------+----
```

-----+--------+---------- -+---------+-------+------+-------+------+----+--------+--------+----+------ +----------
-------+------------------+ only showing top 20 rows Data Preparation
# # Repartitioned the data set to 8 partitions # flights = flights.repartition(8)
flights.rdd.getNumPartitions()
Out[7]:
# Show the inferred schema for the airports dataframe flights.printSchema() root |--
id: integer (nullable = true) |-- year: string (nullable = true) |-- month: string (nullable =
true) |-- day: string (nullable = true) |-- dep_time: string (nullable = true) |--
sched_dep_time: string (nullable = true) |-- dep_delay: string (nullable = true) |--
arr_time: string (nullable = true) |-- sched_arr_delay: integer (nullable = true) |--
arr_delay: string (nullable = true) |-- carrier: string (nullable = true) |-- flight: integer
(nullable = true) |-- tailnum: string (nullable = true) |-- origin: string (nullable = true) |--
dest: string (nullable = true) |-- air_time: string (nullable = true) |-- distance: string
(nullable = true) |-- hour: integer (nullable = true) |-- minute: integer (nullable = true) |--
time_hour: timestamp (nullable = true) |-- name: string (nullable = true)
flights.describe().show() +-------+----------------+------+-----------------+----------------+------ ---------
--+-----------------+-----------------+-----------------+------- ----------+-----------------+-------+-----------------
+-------+------+---- --+-----------------+-----------------+-----------------+---------------- -+------------------+
|summary| id| year| month| day| dep_time| sched_dep_time| dep_delay| arr_time|
sched_arr_delay| arr_delay|carrier| flight|tailnum|origin| dest| air_time| distance| hour|
minute| name| +-------+----------------+------+-----------------+-----------------+------ -----------+-----------
------+-----------------+-----------------+-------+ ----------+-----------------+-------+-----------------+-------+------
+---- --+-----------------+-----------------+-----------------+---------------- -+------------------+ | count|
336776|336776| 336776| 336776| 328521| 336776| 328521| 328063| 336776|
327346| 336776| 336776| 334264|336776|336776| 327346| 336776| 336776|
336776| 336776| | mean| 168387.5|2013.0|
6.548509988835309|15.71078699194717|1349.1099473093045|1344.2548400123524|12.
639070257304708|1502.0549985825894| 1536.380220086942|
6.89537675731489| null|1971.9236198541464| null| null|
null|150.68646019807787|1039.9126036297123|13.180247404803193|
26.23009953203316| null| | stddev|97219.00146576297|
0.0|3.4144572446788914|8.768607101536869|488.28179100116233|
467.3357557342094| 40.21006089212995|
533.2641319903769|497.45714151439483|44.633291690193865|
null|1632.4719381393154| null| null| null| 93.68830465900996|
733.2330333236765| 4.661315707848444|19.300845657412886| null| | min| 0|
2013| 1| 1| 1.0| 1000| -1.0| 1.0| 1| -1.0| 9E| 1| D942DN| EWR| ABQ| 100.0| 1005| 1|
0|AirTran Airways C...| | max| 336775| 2013| 9| 9| 959.0| 959| 99.0| 959.0| 2359| 99.0|
YV| 8500| N9EAMQ| LGA| XNA| 99.0| 997| 23| 59| Virgin America|
To prepare the data for further processing, the null values and the values with empty
strings are dropped.
flights.count()
Out[10]:
flights.na.drop("all")
Out[11]:

DataFrame[id: int, year: string, month: string, day: string, dep_time: string, sched_dep_time: string, dep_delay: string, arr_time: string, sched_arr_delay: int, arr_delay: string, carrier: string, flight: int, tailnum: string, origin: string, dest: string, air_time: string, distance: string, hour: int, minute: int, time_hour: timestamp, name: string]

flights.dropna()

Out[12]:

DataFrame[id: int, year: string, month: string, day: string, dep_time: string, sched_dep_time: string, dep_delay: string, arr_time: string, sched_arr_delay: int, arr_delay: string, carrier: string, flight: int, tailnum: string, origin: string, dest: string, air_time: string, distance: string, hour: int, minute: int, time_hour: timestamp, name: string]

flights = flights.filter((flights.year != '') & (flights.month != '') & (flights.day != '') & (flights.carrier != '') & (flights.origin != '') & (flights.dest != '') & (flights.dep_delay != '') & (flights.arr_time != '') & (flights.air_time != '')) flights.count()

Out[13]:

total_flights = flights.count() unique_flights = flights.dropDuplicates().count()

print("Number of duplicate rows = ",total_flights - unique_flights) Number of duplicate rows = 0

flights = flights.drop('time_hour')

Data Analysis

from pyspark.sql.functions import * import pandas as pd import seaborn as sns import matplotlib.pyplot as plt %matplotlib inline flights_pd = flights.toPandas()

Visualisation

delay_columns = ['dep_delay', 'arr_delay', 'distance'] delay_data = flights_pd[delay_columns] correlation_matrix = delay_data.corr() # Create a heatmap of the correlation matrix plt.figure(figsize=(10, 8)) sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0) plt.title('Correlation Between Flight Delays') plt.show()

delay_columns = ['dep_delay', 'arr_delay'] delay_data = flights_pd[delay_columns] correlation_matrix = delay_data.corr() # Create a heatmap of the correlation matrix plt.figure(figsize=(10, 8)) sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0) plt.title('Correlation Between Flight Delays') plt.show() plt.figure(figsize=(10, 6)) sns.histplot(flights_pd['air_time'], bins=20, kde=True) plt.title('Flight Duration Distribution') plt.xlabel('Flight Duration (minutes)') plt.ylabel('Frequency') plt.grid(True) plt.show()

# Count the number of flights for each airline airline_counts = flights_pd['carrier'].value_counts() # Create a bar plot plt.figure(figsize=(12, 6)) sns.barplot(x=airline_counts.index, y=airline_counts.values) plt.title('Number of Flights by Airline') plt.xlabel('Airline') plt.ylabel('Number of Flights') plt.xticks(rotation=45) plt.tight_layout() plt.show()

flights_pd.dep_delay = pd.to_numeric(flights_pd.dep_delay, errors='coerce')

flights_pd.loc[(flights_pd.dep_delay > 15), 'DepDelay']='Yes'

flights_pd.loc[(flights_pd.dep_delay <= 15), 'DepDelay']='No'

target = flights_pd['DepDelay'].value_counts() plt.pie(target, labels= ["Yes","No"],autopct = "%0.2f",shadow = True,startangle = 140) plt.title = ("DepDelay") plt.legend(loc = "lower left") plt.show()
flights_pd['DepDelay'].value_counts().plot.bar()
Out[22]:
<Axes: xlabel='DepDelay'>
flights_pd.arr_delay = pd.to_numeric(flights_pd.arr_delay, errors='coerce')
flights_pd.loc[(flights_pd.arr_delay > 15), 'ArrDelay']='Yes'
flights_pd.loc[(flights_pd.arr_delay <= 15), 'ArrDelay']='No'
# Calculate average delay for each carrier flights_pd['TotalDelay'] = flights_pd['arr_delay'] + flights_pd['dep_delay'] average_delay_per_carrier = flights_pd.groupby("carrier")["TotalDelay"].mean()
# Create a bar chart plt.figure(figsize=(10, 6)) average_delay_per_carrier.plot(kind="bar", color="skyblue") plt.title = ("Average Flight Delay by Carrier") plt.xlabel("Carrier") plt.ylabel("Average Arrival Delay (minutes)") plt.xticks(rotation=0) plt.tight_layout() plt.show()
target = flights_pd['ArrDelay'].value_counts() plt.pie(target, labels=["Yes","No"],autopct = "%0.2f",shadow = True,startangle = 140) plt.title = ("ArrDelay") plt.legend(loc = "lower left") plt.show() flights_pd['ArrDelay'].value_counts().plot.bar()
Out[24]:
<Axes: xlabel='ArrDelay'>
fig, axes = plt.subplots(3, 1, figsize=(20,15)) plt.subplots_adjust(hspace=0.5) sns.set_theme() axes[1].set_title('Data Distribution by Carriers') sns.histplot(flights_pd['carrier'], bins=40, kde=True, alpha=0.7, ax=axes[0]) axes[1].set_title('Data Distribution by Origins') sns.histplot(flights_pd['origin'], bins=40, kde=True, alpha=0.7, ax=axes[1]) axes[1].set_title('Data Distribution by Destination') sns.histplot(flights_pd['dest'], bins=40, kde=True, alpha=0.7, ax=axes[2])
Out[25]:
<Axes: xlabel='dest', ylabel='Count'>
flights_pd.distance = pd.to_numeric(flights_pd.distance, errors='coerce')
flights_pd.air_time = pd.to_numeric(flights_pd.air_time, errors='coerce') # A 1x2 grid of subplots fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 7), constrained_layout=True) # Subplot 1 sns.boxplot(x='DepDelay', y='distance', data=flights_pd, palette='Set1', ax=axes[0]) axes[0].set_title("Box Plot of Distance vs. Departure Delay", fontsize=12) # Subplot 2 sns.boxplot(x='DepDelay', y='air_time', data=flights_pd, palette='Set1', ax=axes[1]) axes[1].set_title("Box Plot of Air Time vs. Departure Delay", fontsize=12) # Adjust layout with vertical padding plt.tight_layout() # Show the plot plt.show()
C:\Users\suraj\AppData\Local\Temp\ipykernel_5024\3455045790.py:16: UserWarning: The figure layout has changed to tight plt.tight_layout()
Average Flight delays of different Airlines
flights.createOrReplaceTempView("flights") spark.sql(""" SELECT name, CAST(AVG(arr_delay + dep_delay) as DECIMAL(11,2)) AS AvgDelay FROM flights GROUP BY name ORDER BY AvgDelay DESC """).show() +--------------------+--------+ | name|AvgDelay| +--------------------+--------+ |Frontier Airlines...| 42.12| |AirTran Airways

C...| 38.72| |ExpressJet Airlin...| 35.64| | Mesa Airlines Inc.| 34.46| |Southwest Airline...| 27.31| |SkyWest Airlines ...| 24.52| | Endeavor Air Inc.| 23.82| | JetBlue Airways| 22.43| | Envoy Air| 21.22| |United Air Lines ...| 15.57|
| Virgin America| 14.52| |Delta Air Lines Inc.| 10.87| |American Airlines...| 8.93| | US Airways Inc.| 5.87| |Hawaiian Airlines...| -2.01| |Alaska Airlines Inc.| -4.10| +--------------------+--------+

Total Flights per origin airport

spark.sql(""" SELECT origin, COUNT(*) AS TotalFlights FROM flights GROUP BY origin ORDER BY TotalFlights DESC """).show() +------+------------+ |origin|TotalFlights| +------+------------+ | EWR| 117127| | JFK| 109079| | LGA| 101140| +------+------------+

Total Flights per destination airport

spark.sql(""" SELECT dest, COUNT(*) AS TotalFlights FROM flights GROUP BY dest ORDER BY TotalFlights DESC """).show() +----+------------+ |dest|TotalFlights| +----+------------+ | ATL| 16837| | ORD| 16566| | LAX| 16026| | BOS| 15022| | MCO| 13967| | CLT| 13674| | SFO| 13173| | FLL| 11897|
| MIA| 11593| | DCA| 9111| | DTW| 9031| | DFW| 8388| | RDU| 7770| | TPA| 7390| | DEN| 7169| | IAH| 7085| | MSP| 6929| | PBI| 6487| | BNA| 6084| | LAS| 5952| +----+------------+ only showing top 20 rows Average Flight delays according to route

spark.sql(""" SELECT concat(origin, " - ", dest) AS route, CAST(AVG(arr_delay + dep_delay) as DECIMAL(11,2)) AS AvgDelay FROM flights GROUP BY route ORDER BY AvgDelay DESC """).show() +---------+--------+ | route|AvgDelay| +---------+--------+ |EWR - TYS| 82.81| |EWR - CAE| 78.95| |EWR - TUL| 68.55| |EWR - OKC| 59.80| |EWR - JAC| 59.74| |EWR - DSM| 53.04| |EWR - RIC| 52.24| |LGA - CAE| 49.17| |EWR - ROC| 48.64| |EWR - PWM| 48.50| |EWR - MKE| 47.61| |JFK - CVG| 46.44| |LGA - BHM| 46.11| |LGA - SBN| 45.83| |JFK - CMH| 44.91| |EWR - MSN| 44.41| |LGA - MSN| 42.56|
|EWR - DCA| 42.28| |JFK - EGE| 41.97| |EWR - MCI| 41.93| +---------+--------+ only showing top 20 rows

Average Flight delays according to route distance

spark.sql(""" SELECT CASE WHEN distance < 500 then 'Under 500' WHEN distance between 500 and 1000 then '500-1000' WHEN distance between 1000 and 1500 then '1000-1500' WHEN distance between 1500 and 2000 then '1500-2000' ELSE 'Above 2000' END AS DistanceRange, CAST(AVG(arr_delay + dep_delay) as DECIMAL(11,2)) AS AvgDelay FROM flights GROUP BY DistanceRange ORDER BY AvgDelay DESC """).show() +-------------+--------+ |DistanceRange|AvgDelay| +-------------+--------+ | 500-1000| 22.62| | Under 500| 22.56| | 1500-2000| 17.51| | 1000-1500| 17.29| | Above 2000| 12.08| +-------------+--------+

spark.sql(""" SELECT CASE WHEN day = 1 then 'Sunday' WHEN day = 2 then 'Monday' WHEN day = 3 then 'Tuesday' WHEN day = 4 then 'Wednesday' WHEN day = 5 then 'Thursday' WHEN day = 6 then 'Friday' ELSE 'Saturday' END AS WeekDays, CAST(AVG(arr_delay + dep_delay) as DECIMAL(11,2)) AS AvgDelay
FROM flights GROUP BY WeekDays """).show() +---------+--------+ | WeekDays|AvgDelay| +---------+--------+ |Wednesday| 3.98| | Tuesday| 15.23| | Friday| 5.21| | Thursday| 8.29| | Saturday| 21.17| | Monday| 20.86| | Sunday| 21.50| +---------+--------+ Training Model

Finally, casting the fields to appropriate data types.

the prediction column 'arr_delay' is converted to integer value as 1 when delay > 15 and 0 otherwise.
from pyspark.sql.functions import * flights = flights.select((col("year").cast("Int").alias("year")), \ (col("month").cast("Int").alias("month")), \ (col("day").cast("Int").alias("day")), "carrier", "origin", "dest", \ (col("dep_delay").cast("Int").alias("dep_delay")), \ (col("arr_delay").cast("Int").alias("arr_delay"))) flights = flights.select("year", "month", "day", "carrier", "origin", "dest", "arr_delay", \ ((col("dep_delay") > 15).cast("Int").alias("dep_delay_idx")))
# Importing Libraries from pyspark.ml.classification import LogisticRegression from pyspark.ml.feature import VectorAssembler from pyspark.ml.feature import VectorAssembler, StringIndexer, VectorIndexer, MinMaxScaler from pyspark.ml import Pipeline
# Indexing Categorial features with string values #Step 1 strIdx = StringIndexer(inputCol = "carrier", outputCol = "carrierIdx")
#Step 2 originIdx = StringIndexer(inputCol = "origin", outputCol = "originIdx") #Step 3 destIdx = StringIndexer(inputCol = "dest", outputCol = "destIdx") # Combining all feature variables into one single vector #Step 4 categorial_features = VectorAssembler(inputCols = ["carrierIdx", "year", "month", "day", "originIdx", "destIdx"], outputCol="categorial_features") # Indexing the vector #Step 5 categorial_features_idx = VectorIndexer(inputCol = categorial_features.getOutputCol(), outputCol = "categorial_features_idx") # Grouping the departure delay #Step 6 delay_range = VectorAssembler(inputCols = ["arr_delay"], outputCol="delay_range") #Step 7 range_features = MinMaxScaler(inputCol = delay_range.getOutputCol(), outputCol="range_features") #Step 8 all_features = VectorAssembler(inputCols=["categorial_features_idx", "range_features"], outputCol="all_features")
Logistic Regression
#Step 9 lr = LogisticRegression(labelCol="dep_delay_idx",featuresCol="all_features",maxIter=10,regParam=0.3) # defining the pipeline that combines all above processes in a workflow pipeline = Pipeline(stages=[strIdx, originIdx, destIdx, categorial_features, categorial_features_idx, delay_range, range_features, all_features, lr])
Splitting the dataset in 8:2 ratio for training and testing purposes
splits = flights.randomSplit([0.7, 0.3]) train = splits[0] # rename the target variable in the test set to trueLabel test = splits[1].withColumnRenamed("dep_delay_idx", "true_dep_delay_idx") train_rows = train.count() test_rows = test.count() print ("Training rows count:", train_rows, " Testing rows count:", test_rows) Training rows count: 229272 Testing rows count: 98074 Testing Model
# Training the model import timeit start_time = timeit.default_timer() lr_model = pipeline.fit(train) elapsed = timeit.default_timer() - start_time print ("Model training complete in:", elapsed, "secs") Model training complete in: 22.857215499999995 secs
prediction = lr_model.transform(test) predicted = prediction.select("all_features", "prediction", "true_dep_delay_idx") predicted.show(10, truncate=False) +----------

----------+---------+---------------- -+ |all_features |prediction|true_dep_delay_idx| +----------+---------+---------------- -+ | [4.0,0.0,0.0,1.0,0.0,11.0,0.09499263622974963] |0.0 |1 | | [4.0,0.0,0.0,1.0,1.0,39.0,0.07142857142857142] |0.0 |0 | | [4.0,0.0,0.0,1.0,1.0,3.0,0.05596465390279823] |0.0 |0 | | [4.0,0.0,0.0,1.0,1.0,7.0,0.054491899852724596] |0.0 |0 | |[4.0,0.0,0.0,1.0,2.0,11.0,0.0861561119293078] |0.0 |0 | | [4.0,0.0,0.0,1.0,2.0,8.0,0.06480117820324005] |0.0 |1 | | [1.0,0.0,0.0,1.0,0.0,4.0,0.061855670103092786] |0.0 |0 | | [1.0,0.0,0.0,1.0,1.0,58.0,0.054491899852724596]|0.0 |0 | | [1.0,0.0,0.0,1.0,1.0,35.0,0.06332842415316642] |0.0 |0 | | [1.0,0.0,0.0,1.0,1.0,29.0,0.07069219440353461] |0.0 |0 | +----------+---------+---------------- -+ only showing top 10 rows Creation of Confusion Matrix

```
tp = float(predicted.filter("prediction == 1.0 AND true_dep_delay_idx == 1").count())
fp = float(predicted.filter("prediction == 1.0 AND true_dep_delay_idx == 0").count())
tn = float(predicted.filter("prediction == 0.0 AND true_dep_delay_idx == 0").count())
fn = float(predicted.filter("prediction == 0.0 AND true_dep_delay_idx == 1").count())
print("tp :", tp, "fp :", fp, "tn: ", tn, "fn : ", fn) tp : 3175.0 fp : 17.0 tn: 76948.0 fn : 17934.0
```

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator evaluator = BinaryClassificationEvaluator(labelCol='prediction', rawPredictionCol="rawPrediction", metricName="areaUnderROC") auc = evaluator.evaluate(prediction) print ("Area under the ROC curve = ", auc) Area under the ROC curve = 0.9999973123191824
```

```
lr_metrics = spark.createDataFrame([ ("Logistic Regression", "Precision", tp / (tp + fp)), ("Logistic Regression", "Recall", tp / (tp + fn)), ("Logistic Regression", "AUC", auc) ], ["Model", "Metric", "Value"]) lr_metrics.show()
```

+------------------+---------+------------------+ | Model| Metric| Value| +------------------+---------+------------------+ |Logistic Regression|Precision| 0.9946741854636592| |Logistic Regression| Recall|0.15040977781988726| |Logistic Regression| AUC| 0.9999973123191824| +------------------+---------+------------------+ Support Vector Machines

```
from pyspark.ml.classification import LinearSVC svm = LinearSVC(featuresCol="all_features", labelCol="dep_delay_idx", maxIter=50) pipeline = Pipeline(stages=[strIdx, originIdx, destIdx, categorial_features, categorial_features_idx, delay_range, range_features, all_features, svm])
```

```
start_time = timeit.default_timer() svm_model = pipeline.fit(train) elapsed = timeit.default_timer() - start_time print ("Model training complete in:", elapsed, "secs") Model training complete in: 22.307613200000006 secs
```

```
prediction = svm_model.transform(test) predicted = prediction.select("all_features", "prediction", "true_dep_delay_idx") predicted.show(10, truncate=False)
```

----------+---------+---------------- -+ |all_features |prediction|true_dep_delay_idx| +----------+---------+---------------- -+ | [4.0,0.0,0.0,1.0,0.0,11.0,0.09499263622974963] |1.0 |1 | | [4.0,0.0,0.0,1.0,1.0,39.0,0.07142857142857142] |0.0 |0 | | [4.0,0.0,0.0,1.0,1.0,3.0,0.05596465390279823] |0.0 |0 |

|[4.0,0.0,0.0,1.0,1.0,7.0,0.054491899852724596] |0.0 |0 | |
[4.0,0.0,0.0,1.0,2.0,11.0,0.0861561119293078] |0.0 |0 | |
[4.0,0.0,0.0,1.0,2.0,8.0,0.06480117820324005] |0.0 |1 | |
[1.0,0.0,0.0,1.0,0.0,4.0,0.06185567010309286] |0.0 |0 | |
[1.0,0.0,0.0,1.0,1.0,58.0,0.054491899852724596]|0.0 |0 | |
[1.0,0.0,0.0,1.0,1.0,35.0,0.06332842415316642] |0.0 |0 | |
[1.0,0.0,0.0,1.0,1.0,29.0,0.07069219440353461] |0.0 |0 | +----------+---------+---------- -+ only showing top 10 rows

```
tp = float(predicted.filter("prediction == 1.0 AND true_dep_delay_idx == 1").count())
fp = float(predicted.filter("prediction == 1.0 AND true_dep_delay_idx == 0").count())
tn = float(predicted.filter("prediction == 0.0 AND true_dep_delay_idx == 0").count())
fn = float(predicted.filter("prediction == 0.0 AND true_dep_delay_idx == 1").count())
print("tp :", tp, "fp :", fp, "tn: ", tn, "fn : ", fn) tp : 13695.0 fp : 2271.0 tn: 74694.0 fn : 7414.0
```

evaluator = BinaryClassificationEvaluator(labelCol='prediction', rawPredictionCol="rawPrediction", metricName="areaUnderROC") auc = evaluator.evaluate(prediction) print ("Area under the ROC curve = ", auc) Area under the ROC curve = 0.999999859642306

```
svm_metrics = spark.createDataFrame([ ("SVM", "Precision", tp / (tp + fp)), ("SVM", "Recall", tp / (tp + fn)), ("SVM", "AUC", auc) ], ["Model", "Metric", "Value"])
svm_metrics.show() +-----+---------+----------------+
|Model| Metric| Value| +-----+---------+----------------+ |
SVM|Precision|0.857760240511086| | SVM| Recall|0.648775403856175| | SVM|
AUC|0.999999859642306| +-----+---------+----------------+
# Union the two PySpark DataFrames combined_metrics =
lr_metrics.union(svm_metrics) # Pivot the DataFrame to have Metrics as columns
pivoted_metrics = combined_metrics.groupBy("Model").pivot("Metric").agg({"Value": "first"}) # Convert the PySpark DataFrame to a pandas DataFrame results =
pivoted_metrics.toPandas() # Set 'Model' column as the index
results.set_index('Model', inplace=True)
Out[50]:
AUC Precision Recall
Logistic Regression 0.999997 0.994674 0.150410
SVM 1.000000 0.857760 0.648775
# spark.stop()
```