Insights to Funding and Acquisitions of Startups

Lavish Thomas
MSc in Big Data Analytics and Artificial Intelligence, Letterkenny Institute of Technology
lavishthomas@gmail.com

Abstract— This technical project aims to provide insights into the investment of the start-ups based on data from 1995 to 2015 using the various techniques which are provided in the Apache Spark Ecosystem. Initially, data cleaning is done using transformation functions provided by Apache Spark Dataframe. Furthermore, SQL data analysis is performed in which metrics like top categories are derived. Subsequently, the supervised machine learning algorithm of Linear regression is used in order to find the fastest growing sectors and predict the investment amount in each category for the year 2020. Afterwards, based on the growth and stability factors of investments, new stereotypes are derived using unsupervised learning algorithm of K Means. A distribution pie chart is also provided for new stereotypes. Finally, a summary chart is prepared about all the techniques and their use cases used in this project.

Index Terms— Big Data Analytics, Investments, Machine Learning, Apache Spark, Dataframe, Linear Regression, KMeans, Supervised Learning, Unsupervised Learning.

I. Introduction

The investment in start-ups companies is the latest trend which is gaining traction in the modern business world. This new shift in investment paradigm is an outcome of diverse opportunities offered by the start-up ecosystem. But these opportunities do not come without their own risk. Hence, it is very important to understand the trade-off between benefits vs liabilities of these start-ups projects. This project is aimed at providing a few insights into the data collected about various companies, their investment history and the Return Of Investment (ROI). The research questions answered by this project will help investors to choose the start-ups with the best chance of success while reducing the mitigation costs.

II. Motivation and Target Metrics

The financial world thrives upon investment options and success. Therefore, it is very important to make an informed decision to ensure the return of investment is sustainable and enduring. Detailed analysis of historic data using SQL, a trend analysis, forecasting model using regression and stereotype derivation based on unsupervised machine learning is provided as the solution. This solution provides the following insights in particular.

- Which were the growing sectors in the last two decades?
- Which are the most promising sectors for the next decade to invest in (projections for the year 2020)?
- What are the new stereotypes which the sectors be categorised to? What is the distribution of these new stereotypes?

Under the supervision of Dr. Shagufta Henna, Letterkenny Institute of Technology, Letterkenny, CO. Donegal

III. Dataset details

The data is extracted from the crunchdatabase by Tripp [1] which is segregated into several CSV files. Out of which the primary focus will be on three data sets.

- Investments: Factual Data (Transactional Data) about the funding transactions (168648 rows).
- Acquisitions : Factual Data (Transactional Data) about the acquisitions(18970 rows).
- Companies: Dimensional Data (Details) about the companies (66369 rows).

The data is about the investments and acquisitions which happened over the last few decades, but this research will be dealing with the transaction happened after 1995 (last 20 years). The older data is filtered in the data cleansing process. The data consist of missing values and outliers due to wrong entries. There is inconsistency for the casing and spacing of the text data. The datasets have a many to many mappings which are dealt using custom lambda/map functions.

IV. Methodology

.

The analytics process aims to provide insights into the research questions specified in the target section II using the flow depicted in Figure 1. The entire project is divided into 4 major modules which are further drilled down to various steps for implementing various use cases. Initially, the data is loaded to HDFS from the local master node Linux file system. This data is read using the Spark session which is further filtered and cleansed, which constitutes the data pre-processing stage. Afterwards, aggregation functions are applied using Spark SQL, of which, the results are saved back to a persistent hive table. In the third module, this data is read from the default database and a regression model is applied for each sector separately

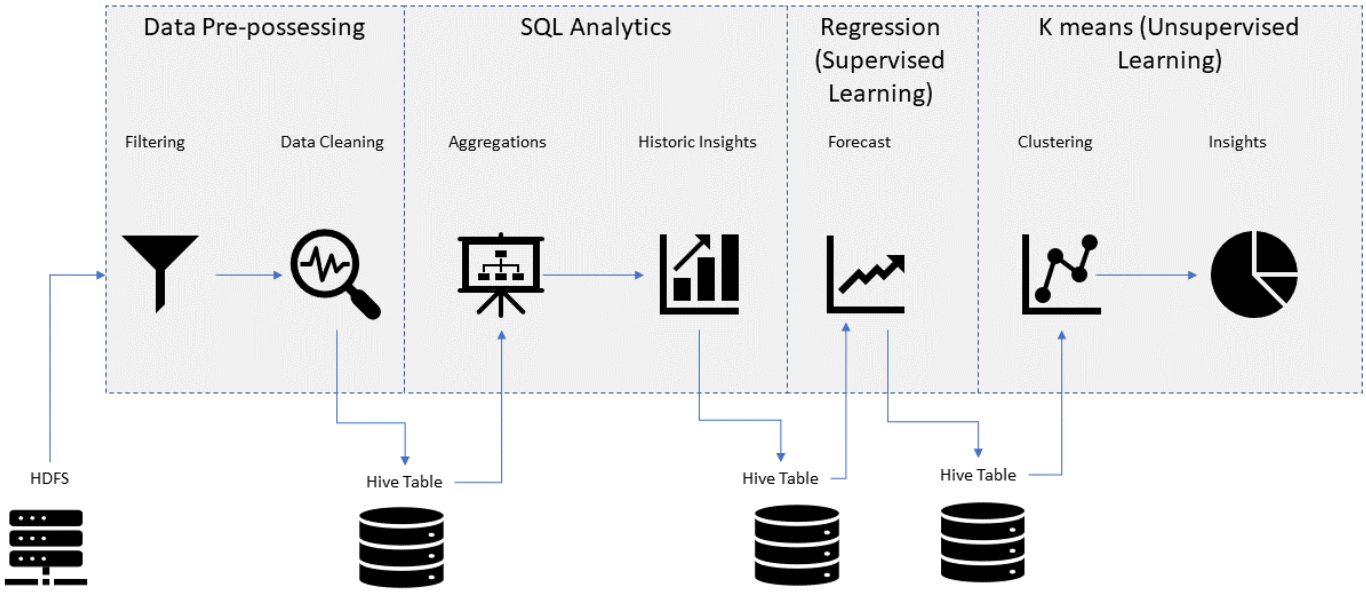

Fig. 1: Analytics pipeline for the investment Analysis

(821 sectors) to forecast the investments for the year 2020. This data is further used to categorise into three new stereotypes named: stable, moderate-risk and high-risk sectors using KMeans.

V. Environment setup

The project is carried out using the following technology stack.

- Local Cluster of 3 Ubuntu (V18.4) Machines [2]
- OpenJDK 11 [3]
- Apache Hadoop 3.2.1 for HDFS storage [4]
- Apache Spark 3.0.0 as processing cluster [5]
- Apache PySpark 2.4.0 and Jupyter Notebooks [6] for coding
- Apache Spark MLib 2.3.0 [7]
- Matplotlib for Visualization [8]

The project is set up in a local machine using the following procedure. Initially Hadoop [4] is installed in the local cluster as per the instruction given in Data Flair blog [9]. Then Spark is installed as per the instruction given in the guide of datawookie [10]. After which the Jupyter Notebook is integrated with the PySpark using the steps described in the tutorial given by Charles Bochet [11]. The data saved in HDFS can be directly accessed by the Spark cluster since the master node of Spark and Hadoop are the same. Putty [12] is used as the ssh client to execute commands in the master node. WinSCP [13] is used for file handling and transfer between the execution cluster and the development machine.

VI. Analytical methods used

This section will explain various techniques which are used in this project.

A. Apache Spark

Apache Spark [14] a cluster-based computing system which works in a distributed manner. APIs in Java, Scala, Python and R are supported in the Spark execution engine. Graph analytics is also supported by the Spark Engine using the module GraphX. The Spark ecosystem consists of complete tool-set for analytics which includes SparkSQL, Spark Streaming, GraphX, MLib.

B. Spark SQL

Spark SQL [15] is one of the major components of the Spark Ecosystem which enable the users to perform data analytics using the queries. This module has gained more traction than other modules because of the indigenous programming expertise using SQL by the data analytics community. The SQL interface also presented the Spark engine to provide more optimization options in the execution using the standard interfaces. The SQL queries can be performed on both temporary tables and persistent tables. There is an option to access external Hive tables using the connectors to the same.

C. Supervised Learning

In the supervised learning systems, the previous decisions, categories and relative metrics are used in order to achieve a model in the form of automated target by providing a different set of values. In other words, these set of values consisting of an input object and a desired or predicted output value [16].

D. Linear Regression

Linear Regression [17] can be used to determine a correlation between parameters which in turn can be used to predict a corresponding value of a parameter,

given the other parameter. As illustrated in the graph below, the data points are approximately residing in a straight line, this property is called a linear correlation between variables. Usually, a "least squares error" is used for finding the line of correlation. In this method, a line is selected which has the minimum sum of least squared error from each point. Gradient descent also can be used which finds the line using the iterative method.

E. Unsupervised Learning

This type of machine learning is used to explore the datasets which have been collected through various sources. This allows for exploring new dimensions in data, unventured correlative paths between variables and clustering of items. One of the main use-cases of this ML type is to find out new categories or to derive new stereotypes [16].

F. KMeans

K-nearest Neighbours [18] is an algorithm which can be used to find categories or to rank similarities based on the distance between data points. The calculation of the distance is usually done an arbitrary function which is specific to the domain and features available in the dataset. Methods such as K-means can be applied, where an iterative method of finding nearby items by calculating the k-centroids for the defined arbitrary value of k as per the use case.

VII. Data Analysis

This section provides a detailed break-up of the analytical procedure used. The process is divided into four major modules, i.e. Data Pre-processing, Spark SQL-based analytics, forecast using regression, clustering using KMeans.

A. Data Pre-processing

This stage comprises of the removing of null values, removing data from irrelevant years etc. The typecasting operations for the various columns are also performed as the CSV to dataframe read operation implicitly considers all fields as a string. Further, the normalization of the data set in terms of structuring is also a major conversion which is handled by this module.

1) Data Loading: The data is first downloaded and extracted into the local node Linux file system. Then, using the Hadoop commands given below, the data is loaded into the HDFS system.

```
hadoop fs -copyFromLocal '/home/lavish/
  startUpFunding/data/* ' '/user/lavish/'data
```

Listing 1: Shell command to load data from Linux system to HDFS

In the PySpark environment, the data is loaded in a spark data frame using the code given below. The dataframe 'iv' is loaded with the data with the schema based on the first row of the dataset.

```
# spark is a custom SparkSession based on some
  config to work with Jupyter notebooks
iv = spark.read.csv("hdfs://localhost:9000/user/
  lavish/data/investments.csv", header='true',
  inferSchema='true')
```

Listing 2: Data Loading in Spark

Similar code is used for loading the other 2 CSV files into the PySpark environment.

2) Data Cleaning : The data is cleaned using dataframe operations. For example, the rows with null values are dropped using below the operation.

```
filteredIV = iv.filter(iv.raised_amount_usd.
  isNotNull())
```

Listing 3: Removing null values in Spark Dataframe operation

3) Data Transformation : Most of the columns are in unnormalized formed, for example, the category of the company is given as 'Curated Web|Games|Analytics' as a single column as shown in the Table I.

Raised amount usd	Funded at	Company category list
8000000.0	04-01-2012	Apps| Cable| Distribution| Software
5.53E7	20-06-2013	Art| E-Commerce| Marketplaces
321650.0	13-11-2006	Local Businesses| Restaurants

TABLE I: Unnormalized data in category field

Two transformation as given below is performed for converting to first normal form to apply the segregation functions later.

First the String column is converted to an array of categories (Table II) using the split operation as shown below.

```
splittedCategoryIV = filteredIV.select('
  raised_amount_usd', substring('funded_at',-4,4)
  .cast('int').alias('year')
, split(col("company_category_list")
, "[|]s*").alias("categoryArr")).filter(col('year'
  ) >= startYear).sort(col("YEAR"))
```

Listing 4: Splitting string to array of categories

Note : the data is filtered in the same statement to reduce the steps for casting and year filtering.

Raised amount usd	Funded at	Company category list
8000000.0	04-01-2012	[Apps, Cable, Distribution, Software]
5.53E7	20-06-2013	[Local Businesses, Restaurants]
321650.0	13-11-2006	[Art, E-Commerce, Marketplaces]

TABLE II: Category column converted into an array using split

Afterwards, the data is converted to first normal form(Table III) using the 'explode' transformation as given below:

```
explodedIV=splittedCategoryIV.select('raised_amount_usd','year', explode('categoryArr').alias('category'))

```

Listing 5: Normalizing the data using explode function

As shown in the table III a row with 3 items in the column array of Category is split into 3 rows.

Raised amount usd	Funded at	Company category list
8000000.0	04-01-2012	Apps
8000000.0	04-01-2012	Cable
8000000.0	04-01-2012	Distribution
8000000.0	04-01-2012	Software
5.53E7	20-06-2013	Local Businesses, Restaurants
5.53E7	20-06-2013	Restaurants

TABLE III: Category column split into multiple rows using explode

B. Top Categories and Trend (Data Analysis)

This section describes the steps to find the top sectors in the past two decades using the provided data. The analysis uses various data frame operations, SparkSQL [19] functions and matplotlib [8] libraries. In brief, the subsections below will explain the various steps in detail about each transformation which are used in this stage.

1) Spark SQL table: The current data frame is registered as a table in order to query and aggregate the dataset based on sectors.

```
explodedIV.write.mode("overwrite").saveAsTable("investments")

```

Listing 6: Saving the data into persistent table

2) Top Categories: To find the top 5 categories, the following query is executed which aggregated based on the category using the 'group by' operation.

```
SELECT CATEGORY, SUM(RAISED_AMOUNT_USD) AS TOTAL,
CAST(SUM(RAISED_AMOUNT_USD) AS DECIMAL(30)) AS TOTAL_DEC
FROM INVESTMENTS GROUP
BY CATEGORY
ORDER BY TOTAL DESC
LIMIT 5

```

Listing 7: SQL Query to find the total amount invested in each sector

The query produces the following results in dataframe format which is converted to array using the following operation.

```
topCategories = [row.CATEGORY for row in categories.collect()]

```

Listing 8: Extracting the categories into an array from a dataframe

The Top categories are:

1) Biotechnology
2) Software
3) E-Commerce
4) Mobile
5) Enterprise Software

These categories are further used to extract the data points for plotting the trend of these sectors.

3) Extracting trend plot points: Using Spark SQL query, the data is aggregated based on categories which will have the total amount invested in each category from the time period 1995 to 2015.

```
SELECT CATEGORY,
CAST(YEAR AS INT),
SUM(RAISED_AMOUNT_USD) AS TOTAL,
CAST(SUM(RAISED_AMOUNT_USD) AS DECIMAL(30)) AS TOTAL_DEC
FROM INVESTMENTS
GROUP BY CATEGORY, YEAR
ORDER BY CATEGORY, YEAR

```

Listing 9: SQL Query to get the data point to plot the trend chart

Note: The data is sorted and type-casted for further calculations.

Then this data is filtered for only the top categories.

```
topCategoriesDF = sqlDF.filter(col('CATEGORY').isin(topCategories))

```

Listing 10: Filtering the data only for the top categories

After which it is loaded to a dictionary format as shown below for feeding to the matplotlib.

```
dict = {}

for row in topCategoriesDF.collect():

if (row.CATEGORY in dict ):
dict[row.CATEGORY]['Y'].append(row.YEAR)
```

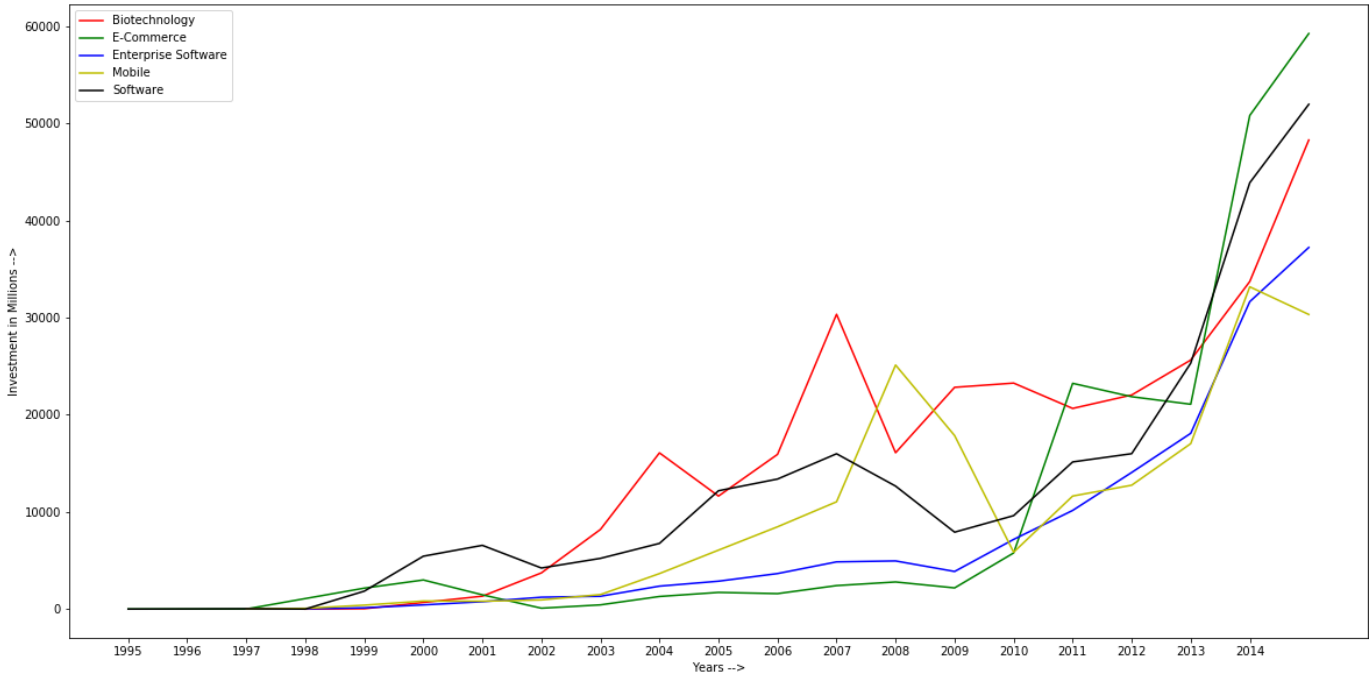

Fig. 2: Trend chart for the top 5 categories

```
    dict[row.CATEGORY]['T'].append(row.TOTAL_DEC
      /1000000)
  else:
  dict[row.CATEGORY] = {  'Y':  [row.YEAR]    ,  'T':[row
    .TOTAL_DEC/1000000]}

```

Listing 11: Populating the dictionary for loading the data into the matplotlib

4) Trend chart for top categories: Based on output of sections VII-B.2 and VII-B.3 the trend chart is plotted as depicted in the figure 2. From the plot, we can infer that most of the sectors have shown steady growth in the past 2 decades. Out of which, E-commerce and Software Industry have gained most in the recent years, closely followed by Bio-Technology and Enterprise Software.

C. Most growing sectors and forecast (Supervised Learning)

Based on the historic data on the investment, supervised learning method the linear regression [20] is used to derive two more metrics:

1) The most growing sectors in last two decades
2) The sectors which will have the highest investments in the year 2020.

1) Loading Data: The prepossessed data from VII-B.1 is loaded in the dataframe using SQL Query given below.

```
SELECT CATEGORY,
CAST(YEAR AS INT),
SUM(RAISED_AMOUNT_USD) AS TOTAL,
CAST(SUM(RAISED_AMOUNT_USD) AS DECIMAL(30)) AS
  TOTAL_DEC
FROM INVESTMENTS
GROUP BY CATEGORY, YEAR
ORDER BY CATEGORY, YEAR
```

Listing 12: SQL Query to get the segregated by category and year

2) Vector Assembler: For applying the machine learning library first vectorAssembler object is created as shown below. The vectorAssembler is used to transform the input columns into feature and target sets.

```
vectorAssembler = VectorAssembler(inputCols = ['
  YEAR'], outputCol = 'FEATURES')
featureDF = vectorAssembler.transform(sqlDF).
  select('CATEGORY', 'FEATURES', 'TOTAL')

```

Listing 13: Vectorization of the dataset

The transformation gives a new dataframe as shown in the Table IV where the features are represented in a single array inside a column named features.

CATEGORY	FEATURES	TOTAL
Mobile Analytics	[2010.0]	4.205E7
Big Data Analytics	[2013.0]	2.35683979E9
Online Scheduling	[2012.0]	9.5505E7

TABLE IV: Vectorized data

3) Prediction points vectorization: The prediction point is set as the year 2020 (5 years to the future from the last data point at 2015). Similar to section VII-C.2, the data is vectorised using below code for feeding to the model transform function later.

```
# Features matrix to predict the amount for the
  Year 2020
```

```
l =  [(2020,)]

rdd = sc.parallelize(l)
test = rdd.map(lambda x: Row(YEAR=x[0] ))
testDF = sqlContext.createDataFrame(test)

vectorAssembler = VectorAssembler(inputCols = ['YEAR'], outputCol = 'FEATURES')
vectorDF = vectorAssembler.transform(testDF).select('FEATURES')
```

Listing 14: Defining and vectorization of the prediction year 2020

4) Model training and prediction: A prediction model is created for each sector, for which the following parameters are calculated:

1) Category
2) Gradient
3) Intercept
4) RMSE
5) R2
6) Prediction

The value of R2 is considered as the accuracy factor for the regression model. The forecasting is discarded for the models which have accuracy less than 50 percentage.

The code for the same is:

```
summaryInfo = []
for category in Categories:
categoryDF=featureDF.filter(featureDF.CATEGORY == category)
if(categoryDF.count() > 10):
lr = LinearRegression(featuresCol = 'FEATURES', labelCol='TOTAL', maxIter=10, regParam=0.3, elasticNetParam=0.8)
lr_model = lr.fit(categoryDF)
if (lr_model.summary.r2 >= .5):
row=(category
,lr_model.coefficients
,lr_model.intercept
,lr_model.summary.rootMeanSquaredError
,lr_model.summary.r2
,lr_model.transform(vectorDF).take(1)[0].prediction )
summaryInfo.append(row)
```

Listing 15: Training of linear regression model for each category

For the further processing of this data in next module, a persistent table is created in the Spark-Hive table as shown below.

```
# Dataframe is saved a SparkSQL Table
columns = ['Category', 'Gradient', 'Intercept','RMSE', 'R2', 'Prediction']
summaryDF = spark.createDataFrame(summaryInfo, columns)
summaryDF.write.mode("overwrite").saveAsTable("summaryDF")
```

Listing 16: Saving the linear regression model details for further processing

5) Regression line plot: The regression line is plotted for the following.

1) Five most growing categories in the past decade
2) Top five categories which will be invested in 2020 according to the prediction.

First, the data is collected from the previously saved data from section VII-C.4 using the following two SQL queries.

```
SELECT CATEGORY, Slope
, Intercept
, R2
, Cast(Prediction as Decimal(12)) as Prediction
FROM summaryDF
ORDER BY Slope Desc
LIMIT 5
```

Listing 17: SQL query to get the top predicted sectors for the year 2020

```
SELECT CATEGORY, Slope
, Intercept
, R2
, Cast(Prediction as Decimal(12)) as Prediction
FROM summaryDF
ORDER BY Prediction Desc
LIMIT 5
```

Listing 18: SQL query to get the top growing sectors

The results are same for both the queries and the sectors are:

1) Biotechnology
2) E-Commerce
3) Software
4) Mobile
5) Enterprise Software

First the data to be plotted is retrieved using the SQL query from the section VII-B.3 based on the these data points the regression line plot as shown in figures 3, 4, 5, 6, 7.

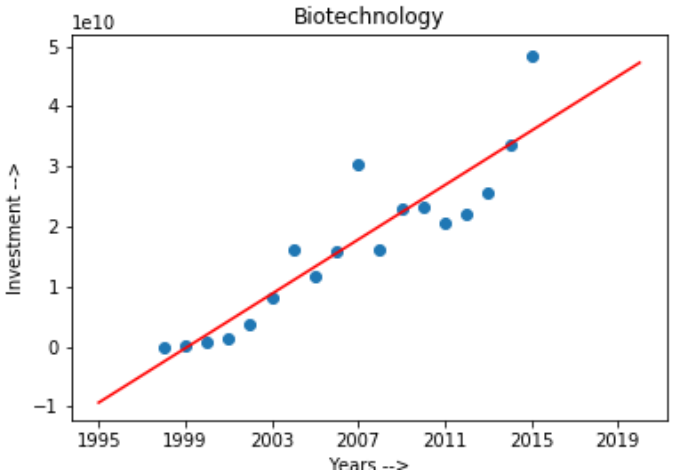

Fig. 3: Regression chart for Biotechnology

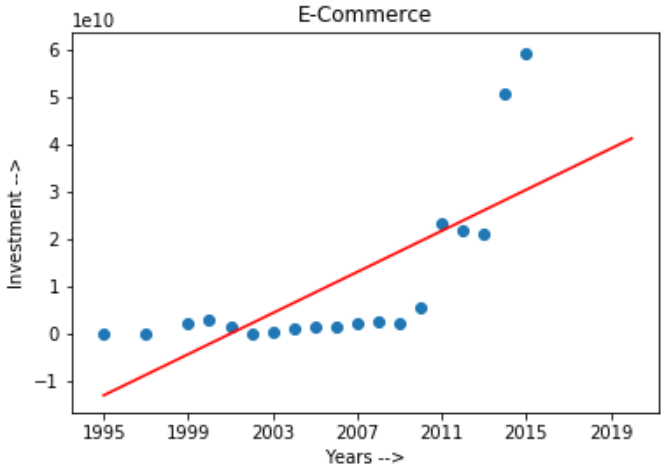

Fig. 4: Regression chart for E-Commerce

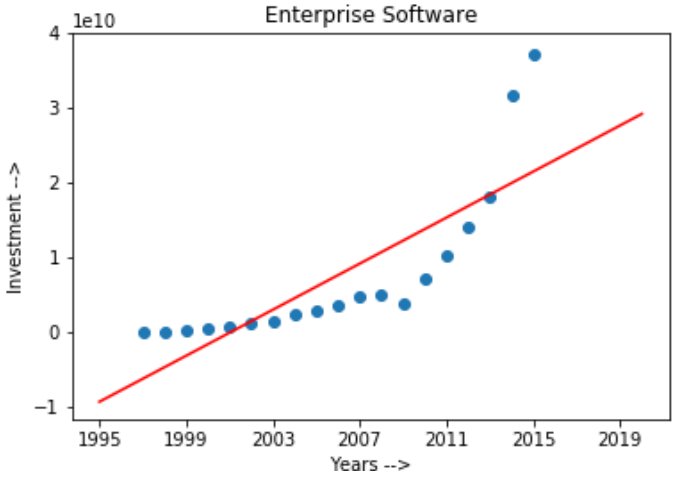

Fig. 7: Regression chart for Enterprise Software

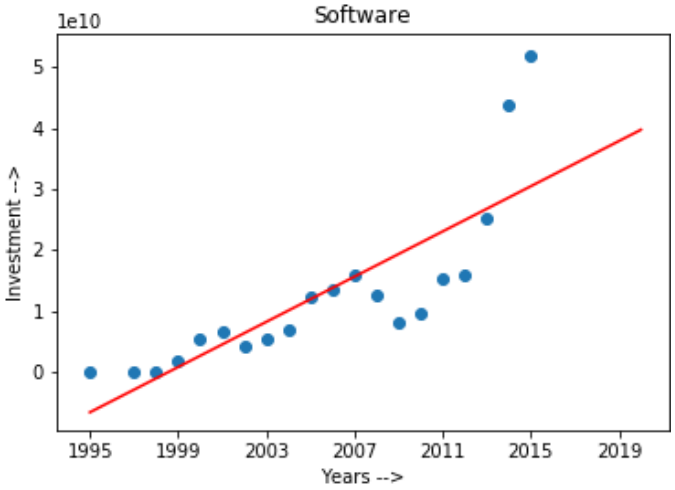

Fig. 5: Regression chart for Software

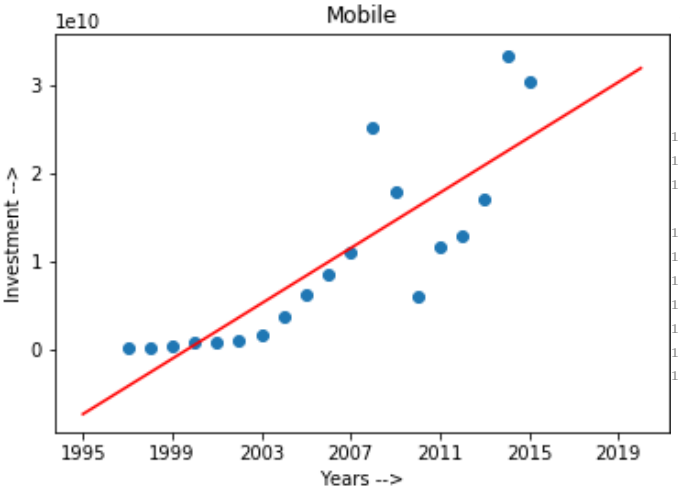

Fig. 6: Regression chart for Mobile

D. Clustering(Unsupervised Learning)

The data we have given is having a pre-set stereotype for categories like software, mobile, health etc which are numerous in number. In this section, based on the growth rate and fluctuation in the growth new stereotypes are derived using unsupervised learning of K-Means [21]. For this section, the data is retrieved from the results of section VII-C which is saved in the persistent Hive table. Afterwards, the data is transformed using the vectorization to feed the K-Means algorithm.

The following code feeds the data into the K Means and retrieves the centroids.

```
# KMeans model is initialized with seed sample 1
  and clustering requirement as 3.
kmeans = KMeans().setK(3).setSeed(1)
model = kmeans.fit(featureDF)

# Assigning the cluster number to each data point
  in the model
predictions = model.transform(featureDF)

# Silhouette score gives confidence of the
  clustering model
evaluator = ClusteringEvaluator()

silhouette = evaluator.evaluate(predictions)
print("Squared euclidean distance and Silhouette
  Value = " + str(silhouette))

# The centroids are retrieved and displayed
centers = model.clusterCenters()
print("Cluster Centers are: ")
for center in centers:
print(center)

```

Listing 19: SQL query to get the top growing sectors

The new centroids are given in the table V.

X value	Y Value
118933605.50119336	0.5948763997431172
1878485079.551565	0.6795320937041847
681651488.7899961	0.6606921165009616

TABLE V: Centroids of the clusters

The clusters of the sector based on the variation and investment amount is as shown below in the fig 8.

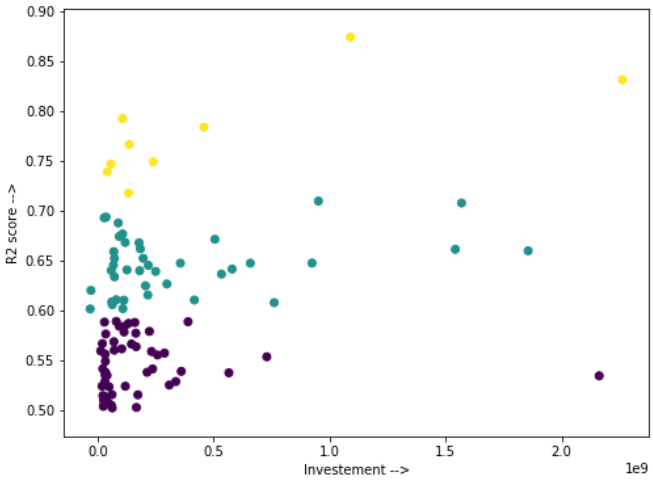

Fig. 8: Cluster of the sectors based on the variation and investment amount

Based on the figure, the clusters can be stereotyped as Fluctuating, Moderate risk, Stable which is further analysed for finding the distribution of the investments in the various types of the risk level. The pie chart is plotted for the same in figure 9

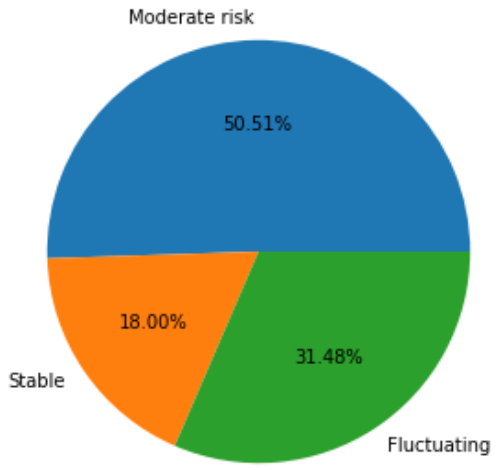

Fig. 9: Distribution of new stereotypes

From the pie chart, it is inferable that the half the companies have a moderate risk level while investing whereas the share of the stable stocks is less in comparison. The risky investment constitutes more than a quarter of the entire portfolio.

VIII. Summary

This data analysis project has taken data about the investments, then given valuable insights to the trend and forecasted the scenarios which will boost the investor confidence. The process has used methods like SQL, regression, clustering which is provided by the Apache Spark environment. Auxiliary technologies like matplotlib, NumPy etc to perform array operations and visualizations. The tools and technologies used are summarized in table VI.

IX. Extension Proposal for the Project

Currently, the data is based on the historic data which is analysed offline in batch processing architecture. This data analytical process is not integrated with the latest investment data which is an ongoing process. Therefore, for enhancement of this system, there are some possible extension paths which are listed below.

1) Integration to the stock market data stream using the spark streaming application which could update the model using the latest data trends for the regression-based forecasting.
2) Creation of a social media and news media data processor which could update the forecasting model using text-based language processor to derive growth potential of each sector.
3) The investment domain has a lot of complex structuring involving the parent company and partial ownership concepts. The GraphX module can be used to solve this complexity and derive more meaningful insights.

X. Best Practices incorporated

1) The data was read directly to a dataframe instead of RDD's
2) Persistent Hive tables were used after expensive operations to save re-computational resource
3) The project is modularized into 4 major modules which constitute the overall flow
4) Pipelines are used in order to streamline the data flow inside the modules
5) Less coupling between modules is achieved using Hive tables as intermediate storage
6) High cohesion of the modules is achieved using the pipeline architecture.

XI. Notes

1) Similar steps for cleaning, filtering and other transformations are no included in the report.
2) The entire code base is available online in the link provide in section XII
3) Notebooks can be executed directly, provided the platform has all the prerequisite libraries explained ins section V

Tools and Techniques	Use case	Target	Result
HDFS	Storing the input files	To make data reading in distributed model	NA
Data frames	Implicit scheme inferring from the csv files	To create dataframe automatically with intended schema	NA
Filters	To clean the null and invalid data	Filtered data from 1995 to 2020 with no null valued rows	Filtered relevant data
Mapper	Transformation of the unnormalized dataframe	To convert the string-based collection of sectors into array	Array based column in dataframe
Explode	Normalized dataframe column representation	To convert the unnormalized data to normalized data	Normalized data
Hive Table	Persistent dataset between major modules	To save intermediate results	NA
Spark SQL	Aggregation functions	Sector wise investment details	Top sectors are identified. VII-B.2
Spark Vectors	Vectorization of the dataset to feed to MLib functions	Vectorized features	NA
Linear Regression (MLib)	Find the trend of the sectors and forecast	1) Top growing sectors 2) Top sector of year 2020	1) Top growing sectors are identified. VII-B.2 2) Forecast for year 2020 is calculated. VII-C.5
K-Means (MLib)	Categorization of sectors	Derive new stereotypes of sectors	Three new sectors (Section VII-D): 1) Fluctuating 2) Moderate risk 3) Stable
Clustering Evaluator	Evaluate the clustering efficiency	Calculate the confidence level of the clustering	77.85%
Matplotlib Scatter Plot	Plot the trends and regression lines	Plot regression charts	NA
Matplotlib Pie Plot	To represent the investment	Visualize the distribution metrics	NA

TABLE VI: Analytics Summary

XII. GitHub Link

The codebase for the project is handled in the GitHub repository given in the below link: https://github.com/lavishthomas/startupFunding.

Folder Description:

1) The Python written in Jupyter are saved in the Notebooks folder of the repository.
2) Various commands to start the services are saved in commands.sh file.
3) Data is saved in the data folder

References

[1] P. Tripp, "notpeter/crunchbase-data," original-date: 2014-08-01T17:01:14Z. [Online]. Available: https://github.com/notpeter/crunchbase-data

[2] Ubuntu 18.04.3 LTS (bionic beaver). [Online]. Available: http://releases.ubuntu.com/18.04/

[3] JDK 11. [Online]. Available: https://openjdk.java.net/projects/jdk/11/

[4] Hadoop – apache hadoop 3.2.1. [Online]. Available: https://hadoop.apache.org/docs/stable/

[5] Preview release of spark 3.0 | apache spark. [Online]. Available: https://spark.apache.org/news/spark-3.0.0-preview.html

[6] Project jupyter. [Online]. Available: https://www.jupyter.org

[7] MLlib | apache spark. [Online]. Available: https://spark.apache.org/mllib/

[8] Matplotlib: Python plotting — matplotlib 3.1.3 documentation. [Online]. Available: https://matplotlib.org/

[9] D. Team. Install hadoop on ubuntu | hadoop installation steps. [Online]. Available: https://data-flair.training/blogs/install-hadoop-on-ubuntu/

[10] A. B. Collier. Installing spark on ubuntu. [Online]. Available: https://datawookie.netlify.com/blog/2017/07/installing-spark-on-ubuntu/

[11] Get started with PySpark and jupyter notebook in 3 minutes. [Online]. Available: https://www.sicara.ai/blog/2017-05-02-get-started-pyspark-jupyter-notebook-3-minutes

[12] Download PuTTY - a free SSH and telnet client for windows. [Online]. Available: https://www.putty.org/

[13] WinSCP. [Online]. Available: https://winscp.net/
[14] Welcome to spark python API docs! — PySpark master documentation. [Online]. Available: https://spark.apache.org/docs/3.0.0-preview/api/python/index.html#
[15] X. Ji, M. Zhao, M. Zhai, and Q. Wu, "Query execution optimization in spark SQL," vol. 2020, pp. 1–12. [Online]. Available: https://www.hindawi.com/journals/sp/2020/6364752/
[16] N. Powell, S. Y. Foo, and M. Weatherspoon, "Supervised and unsupervised methods for stock trend forecasting," in 2008 40th Southeastern Symposium on System Theory (SSST). IEEE, pp. 203–205. [Online]. Available: http://ieeexplore.ieee.org/document/4480220/
[17] U. Ananthakumar and R. Sarkar, "Application of logistic regression in assessing stock performances," in 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech). IEEE, pp. 1242–1247. [Online]. Available: http://ieeexplore.ieee.org/document/8328543/
[18] J. Zhang and K. Yang, "Clustering analysis in the evaluation of securities investment funds," vol. 31, no. 2, pp. 949–956. [Online]. Available: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/JIFS-169024
[19] Spark SQL and DataFrames - spark 2.4.5 documentation. [Online]. Available: https://spark.apache.org/docs/latest/sql-programming-guide.html
[20] Classification and regression - spark 2.4.5 documentation. [Online]. Available: https://spark.apache.org/docs/latest/ml-classification-regression.html#linear-regression
[21] Clustering - spark 2.4.5 documentation. [Online]. Available: https://spark.apache.org/docs/latest/ml-clustering.html