DataSist: A Python-based library for easy data analysis, visualization and modeling

Rising Odegua
Department of Computer Science.
Ambrose Alli University, Ekpoma.
Nigeria.
Email: risingodegua@gmail.com

Abstract

A large amount of data is produced every second from modern information systems such as mobile devices, the world wide web, Internet of Things, social media, etc. Analysis and mining of this massive data requires a lot of advanced tools and techniques. Therefore, big data analytics and mining is currently an active and trending area of research because of the enormous benefits businesses and organizations derive from it. Numerous tools like pandas, numpy, STATA, SPSS, have been created to help analyze and mine these huge outburst of data and some have become so popular and widely used in the field. This paper presents a new python-based library, DataSist, which offers high level, intuitive and easy to use functions, and methods that helps data scientists/analyst to quickly analyze, mine and visualize big data sets. The objectives of this project were to (i) design a python library to aid data analysis process by abstracting low level syntax (ii) increase productivity of data scientist by making them focus on what to do rather than how to do it. This project shows that data analysis can be automated and much faster when we abstract certain functions, and will serve as an important tool in the workflow of data scientists.

Keywords
scientific software, data cleaning, mining, visualization, Python, GitHub

1.0 Introduction

According to ScienceDaily, over 90% of the data in the world was generated in approximately two years [2]. This shows that big data has really come to stay and therefore new research and studies must be carried out in order to fully understand the massive data. This means there has to be a paradigmatic shift from past theories, technologies, techniques and approaches in data mining and analysis in order to fully harness the gold resident in these data [1]. Big data according to [3], has been coined to represent this outburst of massive data that cannot fit into traditional database management tools or data processing applications. These data are available in three different formats such as structured, semi-structured and unstructured format and the sizes are in scales of terabytes and petabytes. Formally, big data is categorized into dimensions in terms of the 3Vs, which are referred to as volume, velocity and variety [8] . Each of the three Vs make traditional operation on big data complicated. For instance, the velocity i.e speed at which the data comes has become so fast that traditional data analytical tools can not handle them properly and may breakdown when used. Also the increase in volume has made the extraction, storage and preprocessing of data more difficult and challenging as both analytical algorithms and system must be scalable in other to cope and these were not built into traditional systems from the onset. Lastly, the ever changing variety of data and its numerous source of integration makes the storage and analysis of data difficult.

The growth of big data has been exponential, and from the perspective of information and communication technology, it holds the key to better and

robust products for businesses and organizations. This outburst as we have stated earlier comes with its own difficulties as regard analysis and mining, and this has been a major hindrance in the massive adoption of big data analytics by many businesses. The major problems here is the lack of effective communication between database systems with the necessary tools such as data mining and statistical tools. These challenges arise when we generally want to discover useful trends and information in data for practical application.

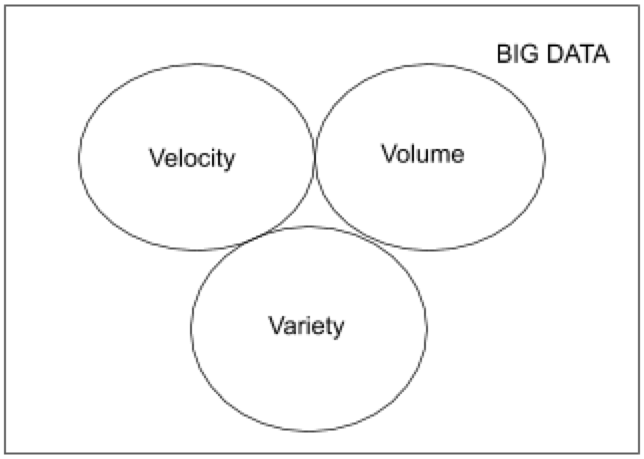


Figure 1:The three Vs of big data

"With big data comes big responsibilities". This is true and is the reason why most accumulated data in industries such as health care, retail, wholesale, scientific research, web based applications etc are dormant and underutilized. In order to fix these problems, we have to implement and understand the numerous ways to analyze big data. Similarly, it has been observed that many analytical techniques built to solve these problems are either too low level to learn easily, too specific to a task, or do not scale to big data. This necessitated for the creation or when possible redesign of existing tools for the sole purpose of solving this problem.

Numerous shortcomings have been identified in most of the Python tools created to solve the challenges mentioned above. Lengthy lines of code for simple tasks, too low level for users, and no support for popular repetitive functions used during analysis are the major problems raised. To solve these challenges, we created DataSist, a Python-based library for easy analysis, data mining and visualization. The functionalities of DataSist is encapsulated in several Python modules which handles different aspects of a data scientist's workflow. DataSist aims to reduce the difficulties in mining data by providing easy, high level, abstracted syntax that are easy to write and remember, and increase productivity and efficiency of the users, by making them focus on what to do rather than how to do it.

2.0 Literature Review

Big data is large; hence the information resident in it must be mined, captured, aggregated and communicated by data scientists/analysts. In order to fully to carry out these tasks effectively, data analysts are expected to have a specific kind of knowledge and to leverage powerful big data analytics tools.

There exists lots of tools for big data mining and these are broadly categorized into three groups, statistical tools, programming languages and visualization tools [10].

Many statistical tools like SPSS, STATA, Statistica etc, are popular for big data analysis but the choice of which to use vary among data analyst as the usage dictates the choice of tool. The SPSS (Statistical Package for the Social Sciences) is widely used tool in the field, though it was originally built for the social sciences [23], SPSS software is now used by market researchers, data miners [24], survey companies, health researchers and others.

STATA is another popular statistical tool used mainly by analysts. One major problem with STATA is that unlike SPSS, it is more difficult to use by people from non-statistical background. STATA is a general purpose tool and is used mainly in the field of economics, sociology, medicine and computer science [25].

Programming languages like Python and R are also famous for data analysis and mining. The high level, dynamic and interactive nature of these languages combined with the abundance of scientific libraries make them a preferred choice for analytical tasks [17]. While R is still the most popular language for traditional data analytical tasks, the usage of Python language has been on the increase since the early 2000s, in both industry applications and research [21]. These has led to the development of the Python ecosystem of libraries, some of which are:

Numpy (Numerical Python), which offers fast and efficient interface for working with high multi-dimensional arrays and matrices. Numpy is a base data structure and fundamental package in Python and as such numerous libraries are also built on top of it.

Pandas is another popular package built on top of Numpy that is used for data manipulation and analysis [19]. Pandas offers efficient data structures and operations for manipulating data usually in tabular forms. Some of the features available in Pandas are DataFrame objects for data manipulation, tools for reading and writing data between memory, hierarchical axis indexing functions to work with high dimensional data, time series functionality, data filtration and data alignment for handling missing data. The library is highly optimized as core modules are written in C and Cython.

Scipy is also another popular library in the Python scientific ecosystem. It is a collection of packages for efficient computation of linear algebra, matrix, special functions and numerous statistical functions.

One last library we will not fail to mention is IPython. IPython is an interactive computing and development environment used mainly by data scientists and analyst for rapid prototyping and exploration of data [16]. IPython is web based usually in the form of a notebook, that offers rich GUI consoles for inline plotting, exploratory analysis, inline coding and markdown text functionalities.

As mentioned earlier, the R programming language is versatile and extremely popular open source language used by data scientists and statisticians [15]. R offers functional based syntax for performing numerous statistical functions and has powerful debugging facilities. It also offers high level and powerful graphical tools [13]. Some of the features that make R a popular choice for analysts is that it has a short and slim syntax, a long and robust list of packages for numerous analytical tasks, availability on numerous OS and platforms and numerous variants for loading and storing data. Some factors that affect adoption is the higher learning curve for people from non-statistical background [11].

An important aspect of big data analysis is visualization, to this end, numerous tools and software has been built to aid effective data visualization. Most of the programming languages like Python and R has their own plotting packages some of which are R's ggplot, Python's matplotlib, seaborn, bokeh, plotly, JavaScript d3.js etc. There has also been massive development of GUI visualization tools, and some popular ones are Tableau, Power Bi, Qlikview, Spotfire and Google Data Studio [5].

3.0 Background

3.1 Data Analysis

Data analysis is the process of inspecting, cleaning, transforming and modeling data with the purpose of discovering useful information, getting actionable insights, supporting decision making and informing conclusions.

Data analysis has multiple approaches including diverse techniques which depends heavily on the problem. This means there is no single or fixed approach to conducting data analysis. In today's world, data analysis plays a crucial role in making decisions and helping businesses operate more efficiently.

Data analysis includes different techniques some which we briefly:

3.1.1 Data Mining: Data mining is a particular type of data analysis that focuses on predictive modeling rather than descriptive modeling. Data mining is an interdisciplinary subfield of computer science and statistics with the ultimate goal of extracting patterns and knowledge from large datasets. It uses numerous methods like machine learning, statistics and database systems [14].

3.1.2 Business Intelligence: Business Intelligence (BI) deals primarily with data concerned with businesses. BI is a combination of technologies and tools used by businesses for analysis of business information. BI helps inform decision making, identify, develop and create strategic business opportunities, and gives businesses a competitive market advantage [12].

In statistical applications, data analysis may be classified into descriptive statistics, confirmatory data analysis, predictive analysis and exploratory data analysis.

3.1.3 Descriptive Statistics: Descriptive statistics provide detailed summaries about observations or sample of data. These statistics could be quantitative, summary statistics like mean, mode, medians, percentiles, max and min etc. or visual, such as graphs and plots. Descriptive statistics form a basic and gives the analyst an intuition into the underlying data set. In business, descriptive statistics helps in summary of many types of data. For example, Marketers and sales personnel may use buyers historical spending and buying patterns by performing simple descriptive analytics on the data in order to make better product decisions. Descriptive statistics may be divided into Univariate and Bivariate and Multivariate analysis.

3.2 Univariate Analysis

Univariate analysis is one of the simplest ways for describing data. The prefix "Uni" means "one", meaning the analysis deals with one feature at a time. This means that when performing Univariate analysis, we do not consider causal relationships among features but instead the main purpose is to describe a single feature. The most popular descriptive statistics found in univariate analysis include central tendency (mean, mode and median) and dispersion (range, variance, maximum, minimum, quartiles and standard deviations.

Using graphs and charts, there are several types of univariate analysis we can perform, some of which are Bar Charts, Histograms, Frequency Polygons and Pie Charts.

3.3 Bivariate Analysis

Bivariate analysis is one of the simplest forms of descriptive statistics. By bivariate analysis, we mean two features compared side by side for possible relationships [4]. The result of bivariate analysis can be used to answer the question of whether a feature "X" depends on another feature "Y", whether there is a causal or linear dependence among these features and whether one can help predict another. Some popular types of bivariate analysis include scatter plots, regression analysis and correlation coefficients.

3.4 Multivariate Analysis

Multivariate analysis is the analysis of three or more features and the relationship among them. It is more complex than both univariate and bivariate analysis. This type of analysis is mostly performed using special tools and software (Pandas, SAS, SPSS etc.), as working with three or more data features manually is infeasible. Multivariate analysis is mostly preferred when the data set under consideration is diverse, and each feature or relation among features is important [6]. Multivariate analysis has applications in numerous domains some which are dimensionality reduction, Clustering, Variable selection, Classification analysis, discrimination analysis and Latent structure discovery.

3.5 Data Processing

Raw data is almost always useless and must be transformed into a usable form. For example, processing may include converting data into structured form (tabular) as most Python analytical tools can work well with this format, cleaning and removal of outliers or useless features, filling or removal of missing values and standardization/normalization of the data values.

3.6 Data Cleaning

Data cleaning is often a part of the data processing phase. Data cleaning comes after data has been processed and organized. In this stage, data received may contain duplicates, redundant features, errors, or be incomplete. This means the data needs to be cleaned for use. Some of the common task done in this stage may include data deduplication, data normalization, feature segmentation, record matching etc.

3.7 Exploratory Data Analysis

Once data has been processed and cleaned, the next phase is usually the exploration phase. In this phase, data scientists/analysts can apply a variety of techniques to understand and analyze the data [7]. The process of exploration can help the analyst to determine which features are important, the relationship among features as well as their inter-dependencies. Exploratory data analysis also helps the analyst to answer questions raised before the exploration begins and can greatly help in reporting and business intelligence.

3.8 Data Visualization

Data visualization is the graphical representation of data. It includes all the processes and techniques involved in the communication of data in pictorial form that aids easy understanding and communication. This communication is achieved through the use of systematic mapping of graphical points to data values. In order to communicate information present in data, data visualization makes use of statistical graphics, charts, plots and info-graphics etc. According to [9], the main goal of data visualization is to communicate information clearly and effectively through graphical means. To convey ideas from big data effectively, both aesthetics, interpretability and functionality must go side by side. The major aim of data visualization is the provision of insights into complex and diverse data set by showing the key-aspects in a more intuitive way.

4.0 Implementation and Architecture

DataSist has been implemented using the Python programming language, and its design is currently centered around five modules (feature_engineering, modeling, structdata, timeseries and visualization). The feature_engineering module contains functions to handle tasks such as cleaning, filling missing, aggregating, counting features in a datasets, the modeling module contains function used for machine learning modeling, structdata module handles all tasks relating to structured data in tabular form, timeseries module handles temporal features such as date-time, timestamps and finally, the visualization module holds numerous functions that aids easy visualization of features in a dataset. The class diagram for the modules are shown in Figure 2.

DataSist has been created for Interactive use in Jupyter Notebook environments and has been integrated into the workflow of data analyst in Python. It builds on a lot of existing libraries like the efficient DataFrame and Series from Pandas, uses Numpy's fast matrices and array functions and heavily depends on Seaborn and Matplotlib for visualization.

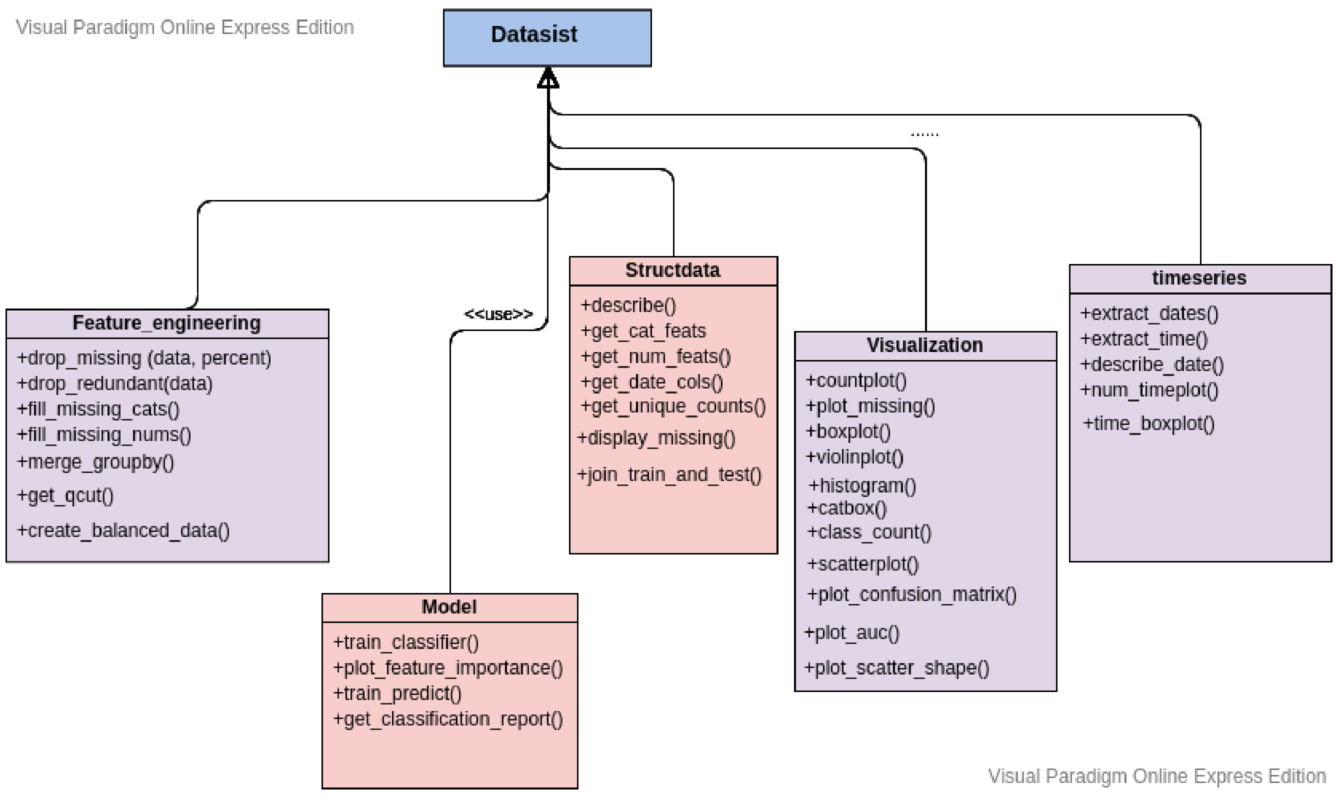
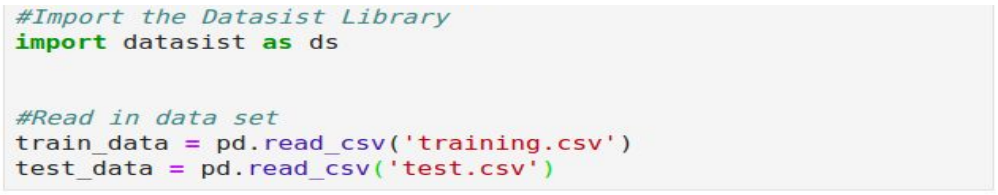


Figure 2: Class diagram showing modules and their member functions

4.1 Use Case

To demonstrate an end-to-end use of the DataSist library, we obtain a dataset from the competitive data science platform Zindi [26]. The dataset is part of a predictive machine learning challenge hosted by Xente [27] an e-payments, e-commerce and financial service company in Uganda. The dataset contains samples of approximately 140,000 transactions between 15th November 2018 and 15th March 2019, and the task is to build a machine learning model from the data to detect if a transaction is fraudulent or not.

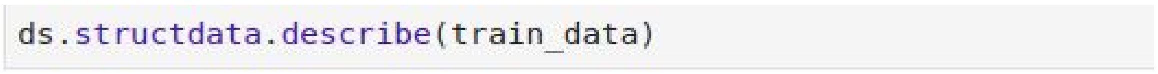
We perform this analysis in an interactive coding environment called Jupyter Notebook [17]. First, we import the necessary libraries to use including the DataSist library.



```
#Import the Datasist Library
import datasist as ds


#Read in data set
train_data = pd.read_csv('training.csv')
test_data = pd.read_csv('test.csv')
```

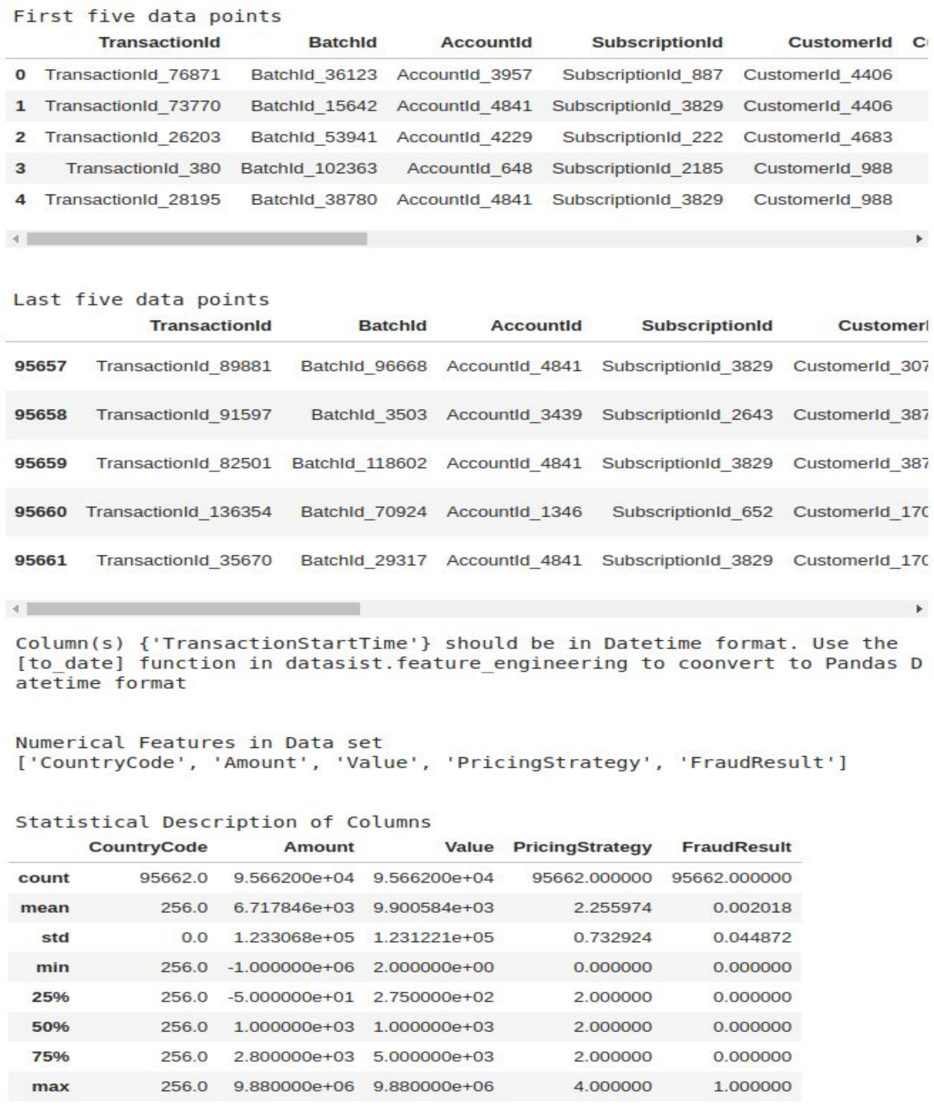
Figure 3: Code importation cell

We can do a quick summary of a dataset using the describe function in the structdata module.



```
ds.structdata.describe(train_data)
```

Result:



```
First five data points
```

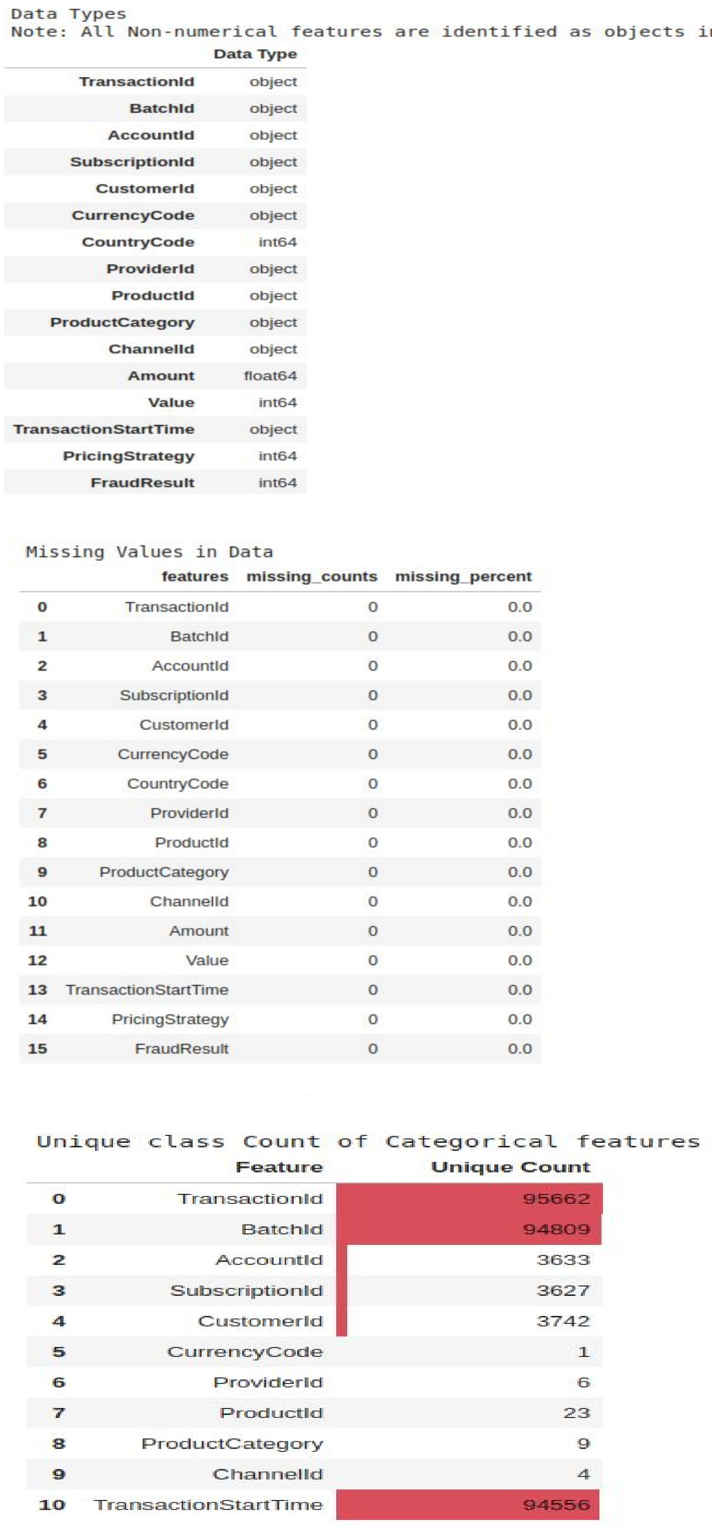
	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId	C
0	TransactionId_76871	BatchId_36123	AccountId_3957	SubscriptionId_887	CustomerId_4406	
1	TransactionId_73770	BatchId_15642	AccountId_4841	SubscriptionId_3829	CustomerId_4406	
2	TransactionId_26203	BatchId_53941	AccountId_4229	SubscriptionId_222	CustomerId_4683	
3	TransactionId_380	BatchId_102363	AccountId_648	SubscriptionId_2185	CustomerId_988	
4	TransactionId_28195	BatchId_38780	AccountId_4841	SubscriptionId_3829	CustomerId_988	

```
Last five data points
```

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerI
95657	TransactionId_89881	BatchId_96668	AccountId_4841	SubscriptionId_3829	CustomerId_307
95658	TransactionId_91597	BatchId_3503	AccountId_3439	SubscriptionId_2643	CustomerId_387
95659	TransactionId_82501	BatchId_118602	AccountId_4841	SubscriptionId_3829	CustomerId_387
95660	TransactionId_136354	BatchId_70924	AccountId_1346	SubscriptionId_652	CustomerId_170
95661	TransactionId_35670	BatchId_29317	AccountId_4841	SubscriptionId_3829	CustomerId_170

```
Column(s) {'TransactionStartTime'} should be in Datetime format. Use the
[to_date] function in datasist.feature_engineering to coonvert to Pandas D
atetime format


Numerical Features in Data set
['CountryCode', 'Amount', 'Value', 'PricingStrategy', 'FraudResult']


Statistical Description of Columns
```

	CountryCode	Amount	Value	PricingStrategy	FraudResult
count	95662.0	9.566200e+04	9.566200e+04	95662.000000	95662.000000
mean	256.0	6.717846e+03	9.900584e+03	2.255974	0.002018
std	0.0	1.233068e+05	1.231221e+05	0.732924	0.044872
min	256.0	-1.000000e+06	2.000000e+00	0.000000	0.000000
25%	256.0	-5.000000e+01	2.750000e+02	2.000000	0.000000
50%	256.0	1.000000e+03	1.000000e+03	2.000000	0.000000
75%	256.0	2.800000e+03	5.000000e+03	2.000000	0.000000
max	256.0	9.880000e+06	9.880000e+06	4.000000	1.000000



```
Data Types
Note: All Non-numerical features are identified as objects in pandas
```

	Data Type
TransactionId	object
BatchId	object
AccountId	object
SubscriptionId	object
CustomerId	object
CurrencyCode	object
CountryCode	int64
ProviderId	object
ProductId	object
ProductCategory	object
ChannelId	object
Amount	float64
Value	int64
TransactionStartTime	object
PricingStrategy	int64
FraudResult	int64

```
Missing Values in Data
```

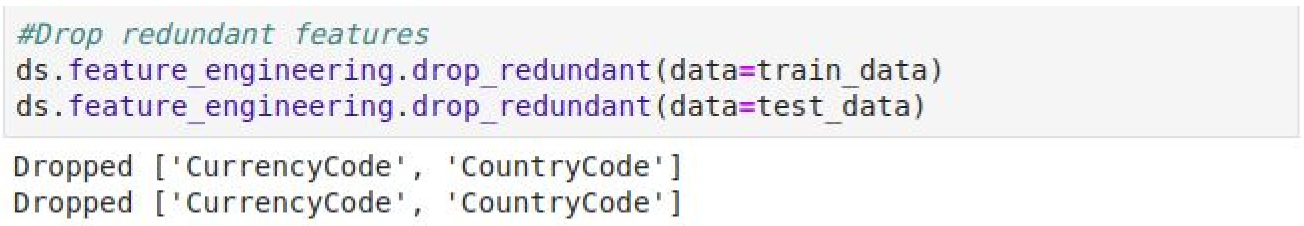
	features	missing_counts	missing_percent
0	TransactionId	0	0.0
1	BatchId	0	0.0
2	AccountId	0	0.0
3	SubscriptionId	0	0.0
4	CustomerId	0	0.0
5	CurrencyCode	0	0.0
6	CountryCode	0	0.0
7	ProviderId	0	0.0
8	ProductId	0	0.0
9	ProductCategory	0	0.0
10	ChannelId	0	0.0
11	Amount	0	0.0
12	Value	0	0.0
13	TransactionStartTime	0	0.0
14	PricingStrategy	0	0.0
15	FraudResult	0	0.0

```
Unique class Count of Categorical features
```

	Feature	Unique Count
0	TransactionId	95662
1	BatchId	94809
2	AccountId	3633
3	SubscriptionId	3627
4	CustomerId	3742
5	CurrencyCode	1
6	ProviderId	6
7	ProductId	23
8	ProductCategory	9
9	ChannelId	4
10	TransactionStartTime	94556

Figure 4: Output from using the describe function on the dataset

Next, we can remove redundant features with the drop_redundant function in the feature_engineering module.

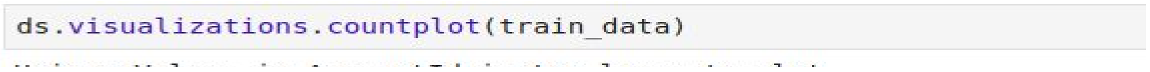


```
#Drop redundant features
ds.feature_engineering.drop_redundant(data=train_data)
ds.feature_engineering.drop_redundant(data=test_data)
```

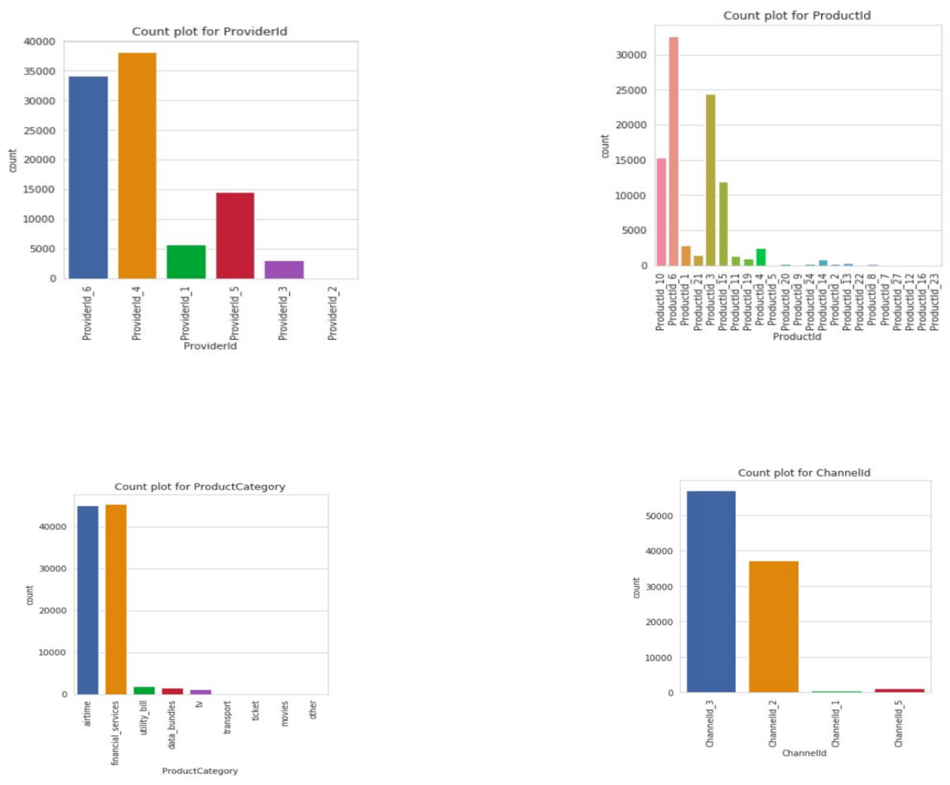
```
Dropped ['CurrencyCode', 'CountryCode']
Dropped ['CurrencyCode', 'CountryCode']
```

Figure 5: Code to drop redundant columns in a dataset

Next, we do some visualization of the categorical features using countplot and catbox in the visualization module. Countplot shows the unique classes in a categorical feature and their corresponding size. This can help us know the most popular classes in a feature. The catplot on the other hand makes a plot of all categorical feature and separates them by a categorical target feature. This is useful in classification task as it helps show which of the classes are important in the separation of the target.

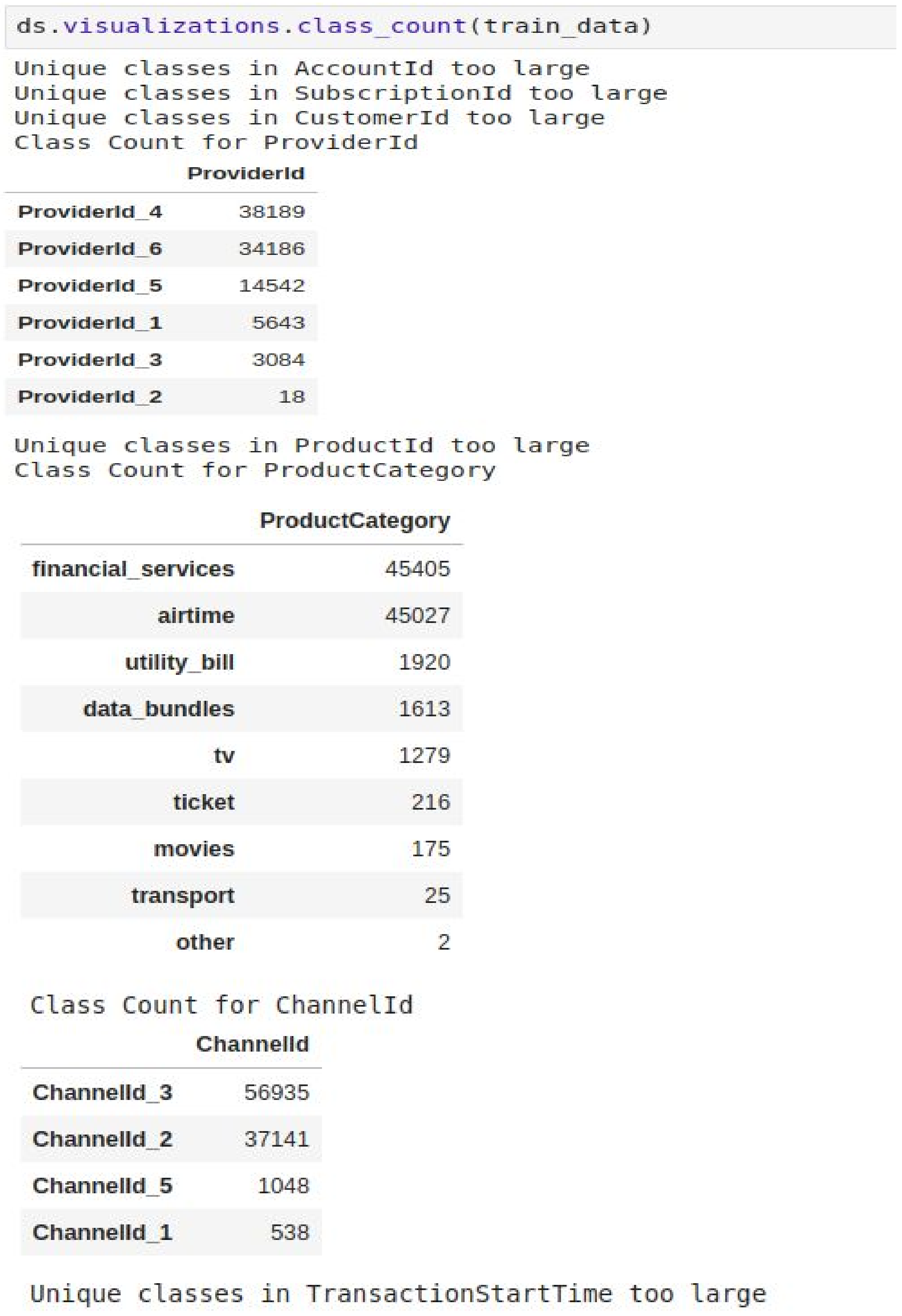


```
ds.visualizations.countplot(train_data)
```



`

Figure 6: Output from visualization of categorical features using countplot



```
ds.visualizations.class_count(train_data)
```

```
Unique classes in AccountId too large
Unique classes in SubscriptionId too large
Unique classes in CustomerId too large
Class Count for ProviderId
```

	ProviderId
ProviderId_4	38189
ProviderId_6	34186
ProviderId_5	14542
ProviderId_1	5643
ProviderId_3	3084
ProviderId_2	18

```
Unique classes in ProductId too large
Class Count for ProductCategory
```

	ProductCategory
financial_services	45405
airtime	45027
utility_bill	1920
data_bundles	1613
tv	1279
ticket	216
movies	175
transport	25
other	2

```
Class Count for ChannelId
```

	ChannelId
ChannelId_3	56935
ChannelId_2	37141
ChannelId_5	1048
ChannelId_1	538

```
Unique classes in TransactionStartTime too large
```

Figure 7: Output from using the class count function

```
ds.visualizations.catbox(data=train_data, target='FraudResult', fig_size=(5,5))
```

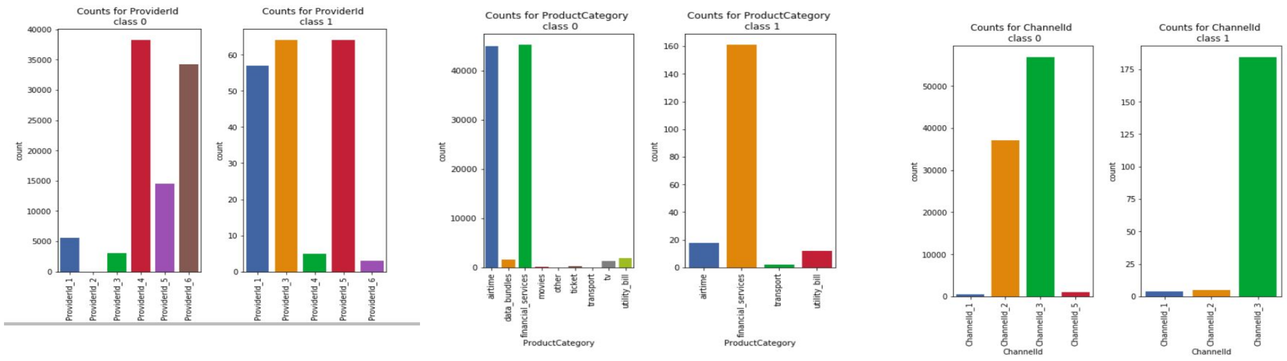


Figure 8: Output from visualization of categorical features using catplot

Next, we can visualize the numerical features in the dataset by using the histogram, boxplot, violinplot or scatterplot also present in the visualization module. The histogram helps to show the distribution of values while the boxplot and violinplot shows the distribution of values based on calculated quartiles such as median, 25th, 50th and 75th percentiles and can help detect outliers.

```
ds.visualizations.histogram(train_data, fig_size=(5,5), bins=5)
```

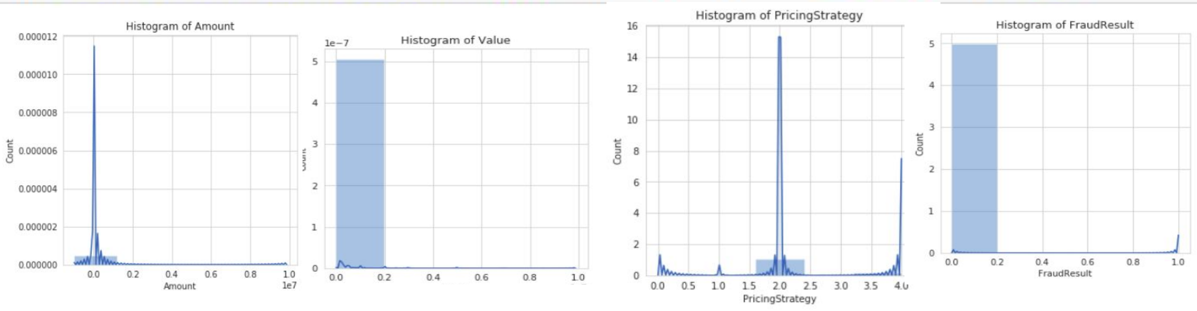


Figure 9: Output from visualization of categorical features using histogram

```
ds.visualizations.boxplot(data=train_data, target='FraudResult', fig_size=(5,5))
```

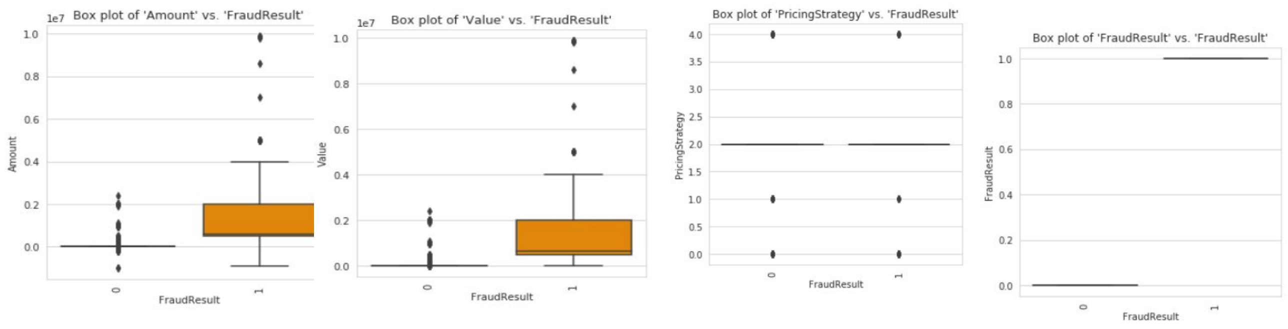


Figure 10: Output from visualization of numerical features using boxplot

Next, we can explore date features/temporal features using the timeseries module. The function num_timeplot can be used to plot all numeric features against time features to check for possible patterns and seasonality, while the extract_dates function can extract date information like hour of the day, minute of the day, day of the month, day of the year, day of the week etc.

```
num_feats = ds.structdata.get_num_feats(train_data)

ds.timeseries.num_timeplot(data=train_data,num_cols=num_feats, time_col='TransactionStartTime')
```

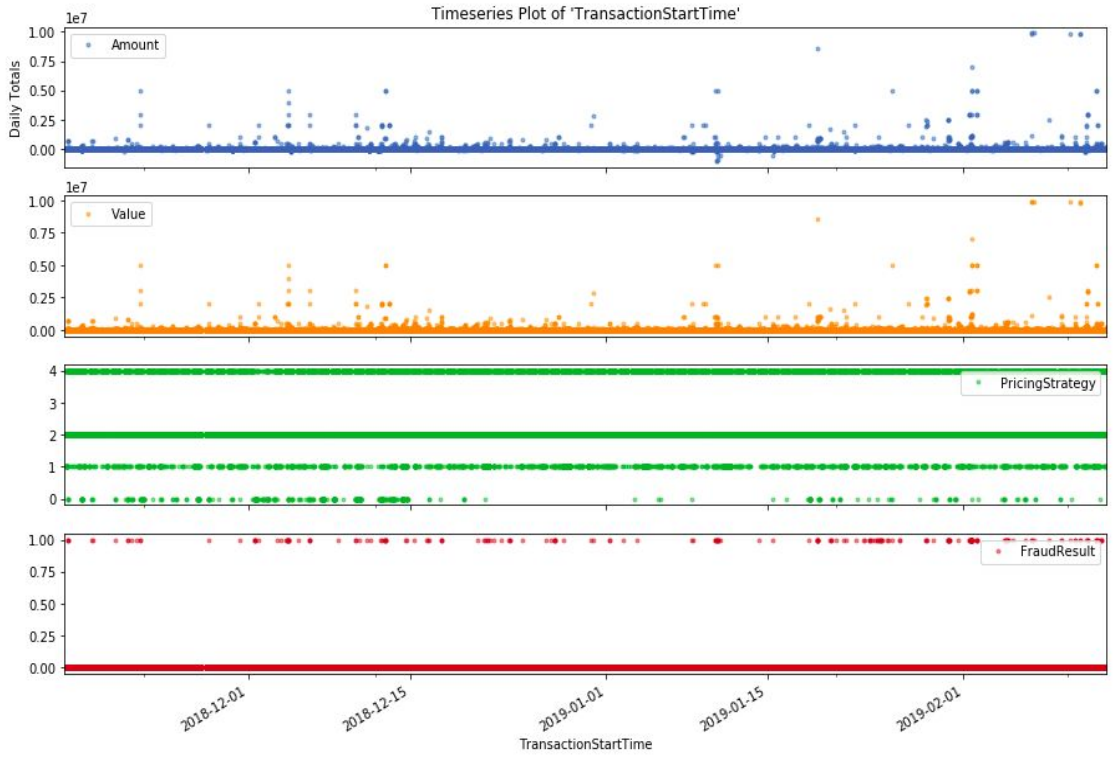


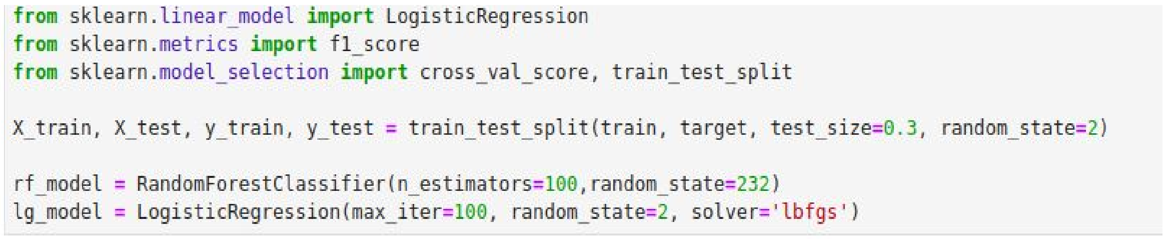
Figure 11: Output from visualization of numerical features against a timestamp feature using the num_timeplot function

```
train_data = ds.timeseries.extract_dates(data=train_data, date_cols=['TransactionStartTime'])
test_data = ds.timeseries.extract_dates(data=test_data, date_cols=['TransactionStartTime'])
```

	0	1
AccountId	AccountId_3957	AccountId_4841
SubscriptionId	SubscriptionId_887	SubscriptionId_3829
CustomerId	CustomerId_4406	CustomerId_4406
ProviderId	ProviderId_6	ProviderId_4
ProductId	ProductId_10	ProductId_6
ProductCategory	airtime	financial_services
ChannelId	ChannelId_3	ChannelId_2
Amount	1000	-20
Value	1000	20
PricingStrategy	2	2
FraudResult	0	0
TransactionStartTime_dow	Thursday	Thursday
TransactionStartTime_doy	319	319
TransactionStartTime_dom	15	15
TransactionStartTime_hr	2	2
TransactionStartTime_min	18	19
TransactionStartTime_is_wkd	0	0
TransactionStartTime_yr	2018	2018
TransactionStartTime_qtr	4	4
TransactionStartTime_mth	11	11

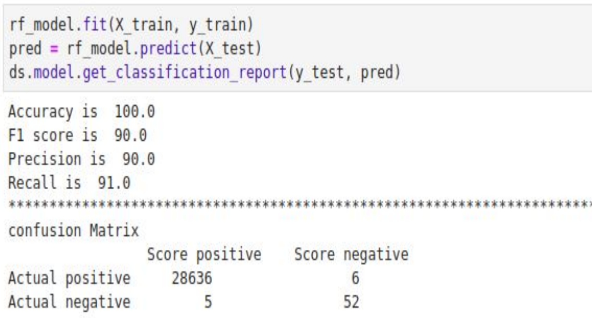
Figure 12: Extraction of date time features using the extract_dates function

Finally, we demonstrate the modeling process of this analysis using modeling module. This module contains functions like train_classifier, get_classification_report and plot_feature_importance that helps in train and test a classification model, generating detailed report from a trained classification model, and plotting a feature importance bar graph respectively.



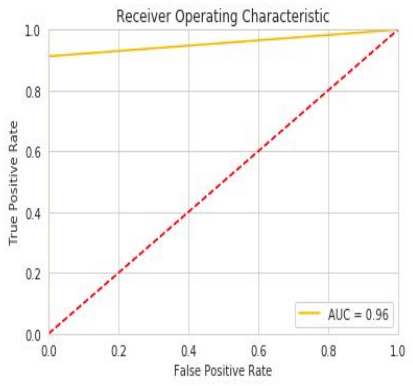
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn.model_selection import cross_val_score, train_test_split

X_train, X_test, y_train, y_test = train_test_split(train, target, test_size=0.3, random_state=2)

rf_model = RandomForestClassifier(n_estimators=100,random_state=232)
lg_model = LogisticRegression(max_iter=100, random_state=2, solver='lbfgs')
```

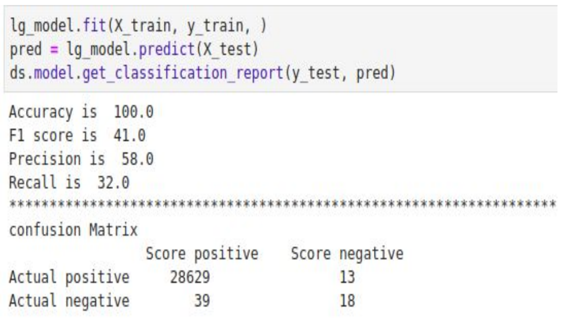
Figure 13: Importation of models and setting up train and test sets



```
rf_model.fit(X_train, y_train)
pred = rf_model.predict(X_test)
ds.model.get_classification_report(y_test, pred)
```

```
Accuracy is  100.0
F1 score is  90.0
Precision is  90.0
Recall is  91.0
**************************************************************************
confusion Matrix
                 Score positive    Score negative
Actual positive     28636                6
Actual negative         5               52
```





```
lg_model.fit(X_train, y_train, )
pred = lg_model.predict(X_test)
ds.model.get_classification_report(y_test, pred)
```

```
Accuracy is  100.0
F1 score is  41.0
Precision is  58.0
Recall is  32.0
**************************************************************************
confusion Matrix
                 Score positive    Score negative
Actual positive     28629               13
Actual negative        39               18
```

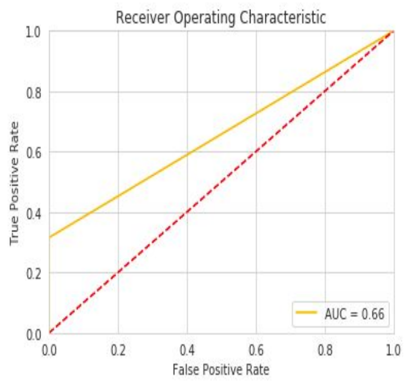
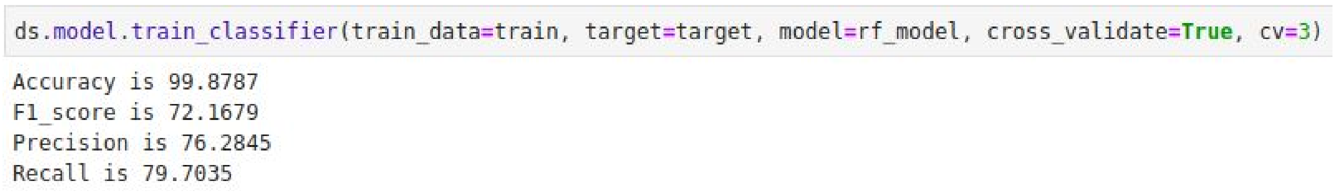


Figure 14: Output from using the get_classification_report function



```
ds.model.train_classifier(train_data=train, target=target, model=rf_model, cross_validate=True, cv=3)
```

```
Accuracy is 99.8787
F1_score is 72.1679
Precision is 76.2845
Recall is 79.7035
```

Figure 15: Metric report from the train_classifier function showing the Accuracy, F1_score, Precision and Recall of the model

```
feats = train.columns
ds.model.plot_feature_importance(estimator=rf_model, col_names=feats)
```

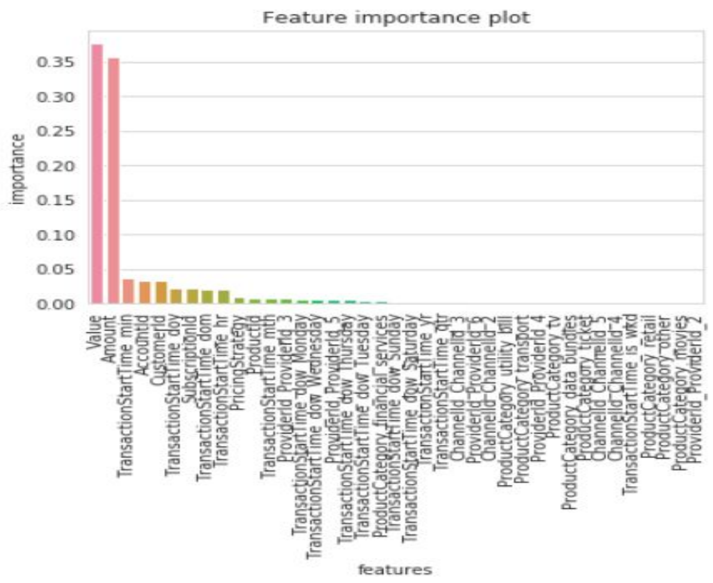


Figure 16: Feature importance plot of the model

4.2 System Requirements and Specification

Operating system

DataSist was tested on Linux(18.04, 19.04 64 bit) and Windows 7,8,10.

Programming language

Python 3.6

Dependencies

- Pandas >= 0.24.2
- Seaborn >= 0.9.0
- matplotlib >= 3.1.0
- numpy >= 1.16.4
- scikit-learn >= 0.21.2

Software Location

Code repository

Name: Github

Identifier: https://github.com/risenW/datasist

Licence: GNU General Public Licence (Version 3)

Language

English

5.0 Conclusion and Future work

In this research work, we show a new Python based data analytical tool called DataSist. Whose aim is to simplify the process of data mining, analysis and visualization and serve as an indispensable tool for data scientists, data analyst and business intelligence experts. We show the basic architecture of DataSist including the underlying modules which are feature_engineering, which contains functions and methods relating to feature cleaning and extraction; modeling, which contains functions that help in building and testing machine learning models; structdata, which contains functions for working with structured tabular datasets; timeseries, which contains temporal related functions and finally visualization module which contains functions for quick visualization of data.

We presented a possible use case of DataSist, by working on dataset from Xente. The task being to create a machine learning model to detect fraudulent transactions. We showed how DataSist can help in quick data summary, feature selection and extraction, extract information from temporal feature (TransactionTime), in visualization relationships between features and finally during modeling.

In the future, we aim to extend DataSist to support advanced workflows in areas such as Deep learning, Natural language processing, Computer vision, Recommender systems, Data reporting etc.This will provide the same easy off-the-shelf tools and techniques for data scientists working in those areas. Overall, we aim to see DataSist as an integrated work tool in the workflow of data scientists and analysts.

References

[1] Caldarola, E. G., Picariello, A., and Castelluccia, D. (2015a). Modern enterprises in the bubble: Why big data matters. ACM SIGSOFT Software Engineering Notes, 40(1):1–4.

[2] Dragland, A. (2013). Big data ? for better or worse.˚ ScienceDaily.

[3] Franks, B. (2012). Taming the big data tidal wave: Finding opportunities in huge data streams with advanced analytics, volume 56. John Wiley & Sons.

[4] Earl R. Babbie, The Practice of Social Research, 12th edition, Wadsworth Publishing, 2009, ISBN 0-495-59841-0, pp. 436–440

[5] J. Stirrup, "Tableau dashboard cookbook", 1st ed., 2014, Packt Publishing, pp.322. [21] S. Redmond, "QlikView for developers cookbook", 1st ed., 2013, Packt Publishing, pp.272.

[6] Olkin, I.; Sampson, A. R. (2001-01-01), Smelser, Neil J.; Baltes, Paul B. (eds.), "Multivariate Analysis: Overview", International Encyclopedia of the Social & Behavioral Sciences, Pergamon, pp. 10240–10247, ISBN 9780080430768, retrieved 2019-09-02

[7] Behrens-Principles and Procedures of Exploratory Data Analysis-American Psychological Association-1997

[8] Mohanty, S., Jagadeesh, M., and Srivatsa, H. (2013). Big Data Imperatives: Enterprise ?Big Data?Warehouse,?BI?Implementations and Analytics. Apress.

[9] Vitaly Friedman (2008) "Data Visualization and Infographics" in: Graphics, Monday Inspiration, January 14th, 2008.

[10] T. Siddiqui and M. Al Kadri, "Big data analytics on the cloud", 2015, International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS), pp. 61–66.

[11] N. Matloff, "Art of r programming", 1st ed., 2011, No Starch Press, Inc., pp.373.

[12] Rud, Olivia. Business Intelligence Success Factors: Tools for Aligning Your Business in the Global Economy, 2009, Hoboken, N.J: Wiley & Sons. ISBN .)

[13] D. Rotolo and L.Leydesdorff, "Matching medline / pubmed data with web of science: a routine in r language", vol. 66, no. 10, 2015, Journal of the Association for Information Science and Technology, pp. 2155–2159.

[14] Clifton, Christopher (2010). "Encyclopædia Britannica: Definition of Data Mining". Retrieved 2010-12-09.
[15] D. Toomey, "R for Data Science", 1st ed., 2014, Packt Publishing, pp.347.
[16] "The IPython notebook: a historical retrospective". Fernando Perez Blog. 8 January 2012.
[17] C. L. P. Chen and C. Zhang, "Data-intensive applications, challenges, techniques and technologies : a survey on big data", vol. 275, 2014, Information Sciences, pp. 314–347.
[18] "Scientific Computing Tools for Python". SciPy.org.
[19] Wes McKinney (2011). "pandas: a Foundational Python Library for Data Analysis and Statistics" (PDF). Retrieved 2 August 2018.
[20] "Python Data Analysis Library – pandas: Python Data Analysis Library". pandas. Retrieved 13 November 2017.
[21] F. Pedregosa, G. Varoquax, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and M. Brucher, "Scikit-learn: machine learning in Python, vol. 12, 2011, Journal of Machine Learning Research, pp. 2825–2830.
[22] IBM, "IBM spss statistics 21 brief guide", 1st ed., 2012, IBM Corp., pp.158.
[23] Nie, Norman H; Bent, Dale H; Hadlai Hull, C (1970). "SPSS: Statistical package for the social sciences".
[24] "KDnuggets Annual Software Poll: Analytics/Data mining software used?". KDnuggets. May 2013.
[25] "Who uses Stata?". Stata. Retrieved 2017-06-28.
[26] Hamilton, Lawrence C. (2013). Statistics with STATA. Boston: Cengage. ISBN .
[27] Zindi, "Data Science Competitions for Africa", https://zindi.africa
[28] Xente, "E-commerce, E-payments and E-solution", /https://www.xente.co/