Credit Card Fraud Detection Using Data Mining and Machine Learning Algorithms

**Abstract**

According to the Legal Information Institute website, credit card fraud is a form of identity theft that involves the unauthorized taking of another person’s credit card information for the purpose of charging purchases to the account or withdrawing funds from it. Machine learning on the other hand can be defined according to the ibm.com website as; a branch of artificial intelligence and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine learning is important as it can be used in the carrying out of repetitive and bulky tasks that require decision making. In the detection of credit card fraud this can be used to beat the manual fraud detection techniques. It improves efficiency and minimizes both time and cost incurred.

This research aims at trying to find a method that beats conventional practices used in credit card fraud detection by making use of machine learning and data analysis. Credit card fraud has been on the rise with the year 2020 having a total of 1,387,615 reports. This was a 44.7% rise from the year 2019. The frauds have taken different forms and this has proven a challenge to the preexisting methods of credit card fraud detection due to their obsolescence and their inability to keep up with the latest creative methods fraudsters use.

From the moment the e-commerce payment systems came to existence, there have always been people who will find new ways to access someone’s finances illegally. This has become a serious problem within the era, as all transactions can easily be completed online by only entering your mastercard information. Even within the 2010s, many American retail website users were the victims of online transaction fraud right before two-step verification was used for shopping online. Organizations, consumers, banks, and merchants are put in danger when a knowledge breach results in monetary theft and ultimately the loss of customers’ loyalty along side the company’s reputation.

Fraud detection can be defined as actions that are taken to recognize attempts to steal money or property through false pretenses, that is identity theft through credit card fraud.  
Frauds are often committed in several ways and in many industries. Most of the detection methods combine various methods of fraud detection and use various datasets and a connected overview of both valid and non-valid payment data to make a choice . This decision may consider attributes of a payment made such as IP address, geo-location, device identification, “BIN” data, global latitude/longitude, historic transaction patterns and transaction information. Fraud can be detected if any of the above attributes fall out of the constraints of what would be considered normal.  
Credit Card Fraud Detection with Machine Learning is a process of investigating a credit fraud by applying the knowledge of a trained machine learning model and revealing and preventing fraudulent transactions. This can be achieved by bringing together all meaningful features of card users’ transactions, like Date, User Zone, Product Category, Amount, Provider, Client’s Behavioral Patterns, etc. The data is then passed through a subtly trained model that finds patterns and rules it can use to classify whether a transaction is fraudulent or is legitimate. All big banks like Chase use fraud monitoring and detection systems.

# **Introduction**

# **Credit Card Fraud**

The four major methods of payment in the world are Cash, Checks, Debit Cards and Credit Cards. According to Statista, a financial services company, in all developing and developed economies, the most used method is the Credit Card, followed closely by the Debit Card.

A credit card is a card – usually plastic - that contains clients’ unique data, often on a magnetic stripe, that approves the individual named on it to pay a merchant for goods and services and be charged occasionally. The data may be accessed by POSs or online payment systems linked to the card. ( Bora et al, 2020).

Fraud, according to Oxford Languages Google Dictionary, is wrongful or criminal deception intended to result in financial or personal gain. As the amount of credit card transactions continues to increase, cases of tricking transaction endpoints to enable fraudsters to use credit cards to fraudulently obtain money or property also continue to increase. This is credit card fraud, which usually boils down to using a credit card belonging to someone else as a source of assets in a given exchange. (Abdou, A. E. H. A. et al, 2019)

Therefore, there is a need to put in place measures to detect these fraud cases. Before, this process was human-intensive. However, with the sharp increase in credit card usage across the globe, more efficient and real-time methods are required.

# **Why Data Mining and Machine Learning**

Machine Learning (ML) is an AI concept that implements the science of using data and creating then applying algorithms that can learn from the past. This technique imitates the way people learn from the data and gradually improves the accuracy of said algorithms. Therefore, it is quite possible to train an algorithm to detect fraudulent transactions while addressing the limitations of the human-intensive process.

By mining big data and being able to work 24/7, ML handles overload well even with the exponential increase in big data size, beating traditional fraud detection techniques.

# **How it works**

In credit card fraud detection, machine learning uses a combination of data mining and modeling to achieve its goal. Simply put, data collection, analysis, segmentation, and extraction processes are succeeded by training a particular model of choice which leads to the creation of the fraud detection system.

The first step involves data collection, analysis, and segmentation before the important data is extracted. This is the data mining stage. Data mining can be described as all the methods and techniques involved in finding correlations and anomalies in huge data sets or big data with the ultimate aim of predicting outcomes. The amount of data an ML model can accept is exponentially larger than what humans can consume, making it superior. Nevertheless, the larger the data set, the better the model can learn and increase precision in fraud detection.

During the extraction stage, the features that can be used to ascertain whether a transaction is normal or otherwise are added. These features may differ due to different conditions like the intricacy of the system, geographical data, or even demographic data.

A training algorithm is used to help the ML model in its decision-making. The amount of data provided for this determines the condition of the ML model. The more the data, the better the model will be.

Once training is done, this whole process results in a model that can detect fraudulent credit card activities. If all conditions under which the model has been produced are optimum, then the model should have the ability to do its job in a very short time.

# **Techniques**

Papers have discussed several ML techniques that can be used toward credit card fraud detection. The techniques, in this case, represent types of algorithms rather than broad categories such as supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning etcetera...

Some of the techniques include Neural Networks, The Decision Tree, Genetic Algorithms, Case-Based Reasoning, Clustering Techniques, Inductive Logic Programming, K-Nearest Neighbor Algorithm, Logistic Regression, Outlier Detection, Support Vector Machine, Bayesian Network among others. (Abdou, A. E. H. A. et al, 2019)

All the above-mentioned techniques have advantages and disadvantages with regards to how they are applied and used. The one I am interested in is the Decision Tree technique.

The Decision Tree technique utilizes a graphical representation of nodes in a tree form with each link between nodes representing decisions. The topmost node is called a root node while the termination node is called a leaf node. This means that each leaf node makes up a classification class while links make up decision paths towards each class. (Goyal & Manjhvar, 2020)

The concept was introduced in the early 90s to handle continuous data. Even though it takes time to check every condition, it has great adaptability, is visually acceptable and it is comparatively simple to produce. (Abdou, A. E. H. A. et al, 2019)

# **Difficulties Involved**

As with any system, certain problems may be encountered in credit card fraud detection efforts. Some of these problems may be advantages over other fraud detection systems but that doesn't mean that they don't affect the system negatively. That said, some of these difficulties can be minimized to maximize the accuracy and precision of the system.

One major difficulty is imbalanced data. This is data that portrays bias towards a class in the data set. Quite naturally, Credit Card Fraud data is usually unbalanced. That means that the algorithm will also be biased toward the same class. (Goyal & Manjhvar, 2020)

Another difficulty especially common with the Decision Tree and other classification models is the lack of adaptability. This means that this algorithm will find it difficult to find new types of normal and/or fraudulent patterns. This is because classification from outdated data will make it ineffective for detecting newer deceptive practices. (Goyal & Manjhvar, 2020)

Lastly, there are no known standard metrics for credit card fraud. There is a lack of evaluation criteria nor is there any means of comparison for fraudulent transactions. Lack of strict standardization hurts the system too. (Goyal & Manjhvar, 2020)

Therefore, in as much as this model makes it easier to carry out fraud detection, these difficulties impede its perfection. The most common coping method is constant improvement and update. And that doesn’t mean algorithmic updates and improvement, but also data. This is because the fraudsters don’t sleep.

The updates may be done either manually or automatically. Manual updates and improvement would mean that each time the model is trained on new data; on the other hand, automatic updates can be carried out through use of big data and unsupervised machine learning, either means may be chosen depending on the the issues the developer needs to overcome. I decided to use supervised learning (which would involve updating the code and training it on new datasets) and will choose a frequency on which to update the model.

**Methodology**

This section helps to review and elaborate on the methods and tools that are put into use in developing a proper and accurate machine learning model. This is dependent on a number of factors that have to be rightly considered and gauged. It involves tools like the choice of a programming language, the choice of libraries and training model. The above mentioned tools coupled with proper data analysis techniques ensure a proper result into a proper and accurate machine learning model.

It would first of all be important to bring to light what a machine learning model is. According to Leonard Eddison in his book, Python Machine Learning , a machine learning model can be described as a file trained to notice patterns in data and make decisions based on those patterns; the decision made becomes the prediction. To develop a model one requires a good machine learning library, a data science library, a data set and an appropriate programming language. There are various machine learning libraries that are free to use and that provide various useful tools that are instrumental in the creation of machine learning models. These libraries have different implementations in different programming languages; it all depends on the user’s preference, my library of choice was the SCIKIT library which has a number of useful functions and classes that can be used in the creation of models. Secondly I had to choose a programming language. This was easy as I needed a language with data manipulation libraries and python proved a worthy candidate. Python is a high level programming language developed by Guido Van Rossum in the late 1980s, at the Centrum Wiskunde & Informatica, as a successor to the ABC programming language. Python is an interpreted language with an easy to write syntax and a robust library, consisting of the pandas module and numpy module, that come in handy for tasks such as data manipulation. Python was therefore my language of choice.

After selecting the various libraries and language for training my model I only had to pick a data set. In machine learning, the ability of the model to make accurate predictions depends on the amount of data used to train the model. By exposing the model to a wider variety of data we increase its level of accuracy. Their are various means through which one can obtain the data for training a model. There is the use of data mining or downloading free data from second party sources. Whichever the way, the obtained data should be cleaned to remove any patterns that may lead to biased predictions by the model. I opted for the second option of obtaining data.

There are various matters to be considered during the selection of a data set for use on training a machine learning model. One should ask questions like; what types of inputs will be used in trying to detect the frauds, what machine learning algorithms will be used to train the model, the viability of the data set chosen based on attributes like; its currentness and its sources. Its best to use the latest data sets that are made up of real life transactions. The choice of data a data set should contain, depends on the developer’s preference; what he deems proper to use in his course of fraud detection. One person may choose to use geo-location and user transaction behavior, while another may choose to use user transaction behavior and the average total amount of expenditure; this is owed to the fact that there is lack of standard means through which credit card frauds can be detected. Either way it is still possible for one to use methods of their own devising and create a model to accurately predict a fraud attempt.

The selection of an algorithm is mostly dependent on the type of data that is used to train the model which will also result in the determination of the data it uses in the prediction of the frauds. Most credit card data, useful to this course, are present in the form of numerical data and this makes it easy as there is no prior conversion of textual data into numerical data and a random sampling all which simplifies the work by a great extent.

The selection of a proper training algorithm and data set effectively contribute to all if not most of the overall model accuracy. The remaining part of the accuracy may be contributed to by the processing and restructuring of the data set and making use of best practices.

A brief elaborate outline of the process involved in creating the model would therefore be; obtaining a suitable data set; one that contains the types of data you would use in detecting credit card fraud, processing and structuring the data set; which means to remove unwanted patterns that would negatively influence the overall accuracy of the trained model, choosing a suitable machine learning algorithm that suit the needs of the conceptual model one aims to end up with and finally saving the trained model and deploying it to a site where it can be accessed and used.

**STATEMENT OF THE PROBLEM**

This project aims to build a machine learning model that can be used in the detection of fraudulent transactions. It aims at finding the best possible method through which fraud can be detected by use of self-reliant programs without human intervention. Currently there are no standard methods through which fraudulent transactions can be determined and there are a variety of ways through which this can be determined. This research project aims to find the best possible way through which standard methods can be developed for the very purpose of fraud detection. Each new day means new ways of committing credit card frauds; the success of detecting future frauds depends on the development of models that will stand the test of time.

**OBJECTIVES**

The main objective of this research was to unveil the best possible way through which machine learning models can be trained to aid in the automatic detection of credit card frauds. It also seeks to unveil the best parameters that can be used in the detection of the fraudulent transactions as currently there is no standard way through which fraudulent transactions can be detected. The realization of standard methods and parameters will aid in ensuring that the technology developed can stand the test of time and can receive a backing from a large and consistent community. Therefore the realization of a long lasting solution that improves over time cam therefore also be considered a major objective of this research project.

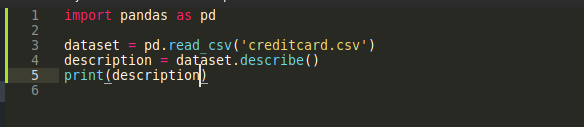
**Data collection and analysis**

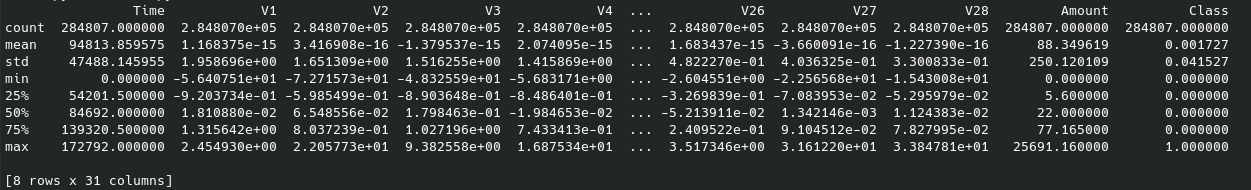
**Describing and visualizing the dataset**

I obtained a data set from kaggle.com which is a web application that hosts a community of data scientists and computer enthusiasts, who share data sets and work collaboratively on different machine learning projects. To start analyzing any data set; it is important for one to have a proper understanding and visualization of the data they are working on. This helps in understanding the data set to help in structuring it to fit a certain training model for the machine learning algorithm employed.

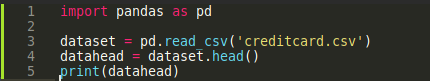
To gain a proper understanding of the data set; the pandas python library offers methods that can be used to read CSV files and perform different operations on the data.

When performing operations on a data set named ‘credit.csv’; the following code would be used to help in visualizing the data.

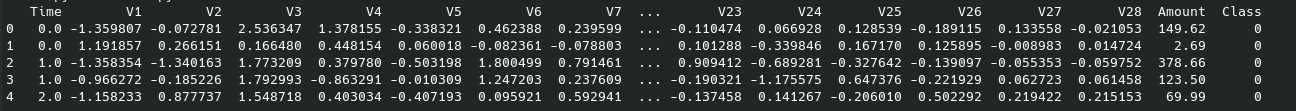
First, one would need to import the pandas python library as shown in line 1 in the image above. The pandas data frame can then be created using the pandas.read\_csv() method which reads the CSV file and then returns a pandas data frame. The pandas data frame is then stored in a variable called ‘dataset’. We can then access and describe the pandas dataframe through the pandas.describe() method; which is then stored in the variable; ‘description’. The python print method can then be used to display the output on a terminal as shown below.

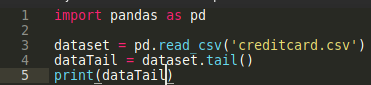
From the output above we can deduce the shape and size of our data; that is in terms of columns and rows. Alternatively one can use the pandas head() and tail() methods as shown below to describe the first five entries and last five entries respectively. It can be done as shown below.

Below is the pandas head() method.

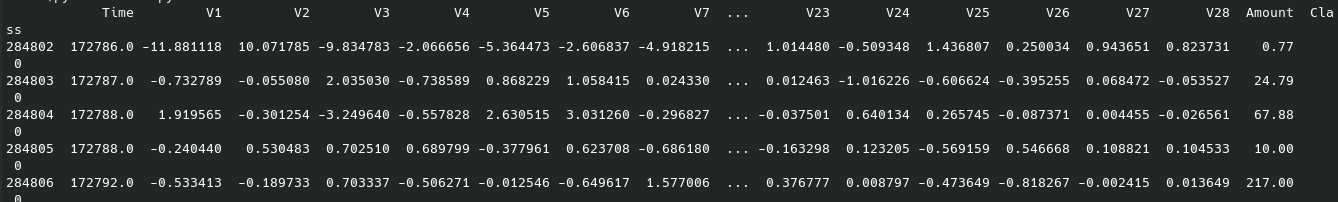


This displays the first five entries.

Below is the pandas tail() method.



This displays the last five entries

There are various pandas library methods that can be used to describe data depending on one’s needs; the above mentioned methods are just an example.

Clearly understanding the data set is as important as understanding each individual column within the data set. This can mean getting the number of rows within each column and knowing whether a column maybe containing any null values. To get the number of columns it would be easier to use the pandas.describe() method which returns the shape of the data set as shown below.



To get the number of rows within the data set it would be better to use the pandas.tail() method which returns the last five entries within the data set. We can know the number of rows by checking the id of the last entry as shown below.

From the above image we can deduce that the data set has 2,848,060 rows; which is a good amount of data to use when training a model.

One may also choose to access each row individually. This is important has it offers access to methods that can be used to check for null values within a data set. Such methods are only accessible for use on single columns within whole data frames. Checking for null values is important as it aids in getting rid of any form of bias that may arise from the absence of such values. The pandas.isnull().sum() method returns the number of null values within the data set. If absent it returns a zero.



The credit card fraud data set used within this data set contained no null values and therefore I carried with further data processing. The next operation on the data set was to check whether the data set was well balanced; this operation was carried out on the ‘class’ column, which had a possible of only two values; that is either 0- to indicate no fraud or 1- to indicate a fraud.

This is possible through the use of pandas.value\_counts() method. The syntax of the method is- dataFrameName[‘column’].value\_counts(). This line of code displays the following outputs.

The code



The terminal output



From the above terminal output it can be easily deduced that the data within our data set is imbalanced and there is therefore need to carry out operations. Using the above data set as it is may lead to a biased model that would have likelihood of predicting a case as not a fraud because that covers over 90% of the total data set. To balance the data set it would be necessary to restructure it, and whereas there are many methods of getting this done, I opted for randomly selecting and mixing the values to obtain a new and more balanced data set.

This can be carried out by first creating two new data sets. This would be done by assigning the class column to two variables depending on whether a value of class was 0 or 1; consequently ending up with two data frames, one whose ‘class’ column values are only 1 and another whose ‘class’ column values are only 0.

That can be carried out as shown in the images below



# We now end up with two new datasets, one representing a legal transaction data set and another representing a fraud transaction data set. To balance the data set we have to randomly pick 492 values from the data frame representing the legal transactions to balance the 492 values from the data frame representing the fraudulent transactions. This can be done by using the pandas.sample() method as shown below.



# The method takes an argument ‘n’ which is an integer value that dictates the number of values that need to be sampled which in our case is 492. The method returns a data frame which is then stored in a variable called ‘legal\_sample’. After creating a randomized data set we join it to the data frame representing fraudulent transactions to create a new balanced data set. This is possible through the pandas.concat() method which takes the form of the following syntax; pandas.concat([dataframe\_A, dataframe\_B], axis=value). It takes two arguments. The first being a list of the data frames one wishes to join and the second is an axis argument which can either be a 0 or a 1. The axis argument helps dictate how the two data frames are joined. And this could be one data frame after the other, or one value from a data frame after another value from another data frame in an alternating manner. This can be done as shown below.

* 

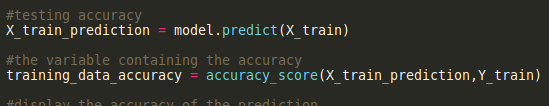
# Having created a balanced data set we can now split our data into two sets; X and Y. One set will be used as labels and another will act as values. The labels act as the prediction and the values act as the key attributes used in determining the prediction. Splitting the data into labels and values is made easier by using the pandas data frames we had created earlier on. This can be done as shown below.The X used to represent the values is created by dropping the ‘Class’ column from the ‘legal\_fraud\_dataset’ created from joining the ‘fraud\_dataset’ to the ‘legal\_dataset’. The Y used to represent the labels is created from accessing only the ‘Class’ column from the ‘legal\_fraud\_dataset’(it contains balanced values for both the legal and fraudulent transactions hence the name). Having done this there is now need to further split the newly formed X and Y data sets further into training and testing portions; this still forms part of the data segmentation. The training set can be thought of as a portion of the data set used to train the model while the testing set can be thought of as the part of the data used in testing for the accuracy of the trained model as will be seen later on. It is important to carry out this practice as it makes it easier to test the accuracy of the model and also ensures that the produced model is not biased. To do this the sklearn.model\_selection.train\_test\_split() method can be used. It takes in multiple arguments the most important being; the two data frames that need to be split for training, the size of the testing data frame that is to be produced and a random state that dictates how the data is to split into training and testing segments. That can be done as shown below.

# This splits the data from the X and Y data frames into four segments; ‘X\_train’, ‘X\_test’, ‘Y\_train’ and ‘Y\_test’, with the test data size taking 20 percent of both the X and Y data frames and the training size taking 80 percent of both the X and Y data frames. The size for segmenting the data depends on the developers preference; but it is considered a good practice to give out 20 percent to testing and 80 percent to training as a way of improving the training, testing and accuracy of the model. It is also best practice to allocate a larger percentage of the data to training as this improves the accuracy of the model.

# Having split the data we can now choose our algorithm and train our model. The python ‘sklearn’ module has a variety of options to choose from but since I had decided to use a decision tree I will choose a training algorithm that has the same. For that matter it would be important to consider the sklearn.tree.DecisionTreeClassifier() method which is a class that has functions that can be used to train our model. After making the necessary imports we can instantiate the model as a variable and then pass the necessary arguments to train our model. That can be done as shown below.



# The DecisionTreeClassifier() class has a fit() function that takes the X\_train and Y\_train data frames as arguments and uses them to train the data. After training the model we can now use the test data to test for its accuracy. This is done by first using the trained model and the testing data to make a prediction and comparing the results of the prediction against the training data. The Python Scikit library provides the sklearn.metrics.accuracy\_score() method that can be used to test for the accuracy of the model. It can be used as shown in the image below.



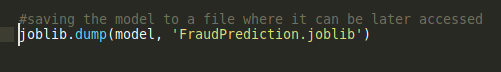
# The accuracy of the model ranges from 0 to 1 with 0 being the lowest level of accuracy any model could have and 1 being the highest. A model may have different levels of accuracy if the code is run more than once but the values should not deviate from a mean value by a large variance. A model with a 0.7 accuracy and above is considered mostly accurate; on the other hand any value below that may not make any effective or reliable predictions. We can view the accuracy of our model by using the python print function to display the output in the terminal. That can be done as shown below.

* 

# The result on the terminal is as shown below.

* 

# My model has an accuracy score of 1.0 which means it has an accuracy of 100%. Having successfully built my model I can now save it to a file and deploy it for access and use. There are various methods that one can use to save a file; python has both the ‘joblib’ and ‘pickle’ libraries for this method. The joblib and pickle libraries help to write to files in different formats and to load and access the contents of the files later on. The pickle library is used to create pickle files, while the joblib library is used create the joblib files. My library of choice was the joblib library. The joblib library has a dump() function that is used to write the model to a file with a joblib extension. The code below shows how that can be done.

****

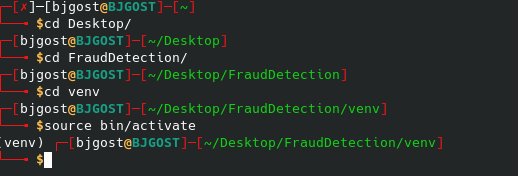
# The code saves our model to a file called ‘FraudPrediction.joblib’; from here it can be accessed and used multiple times.

# **DEPLOYING THE MODEL**

# There are various ways to go about this; and it all depends on the developers preferences. My preference was to deploy the model by means of an application programming interphase or API. According to mulesoft.com an API is a software intermediary that allows two applications to communicate with each other. APIs can be developed in various ways with different programming languages. My language of choice was python; and python has two frameworks through which APIs can be developed. That is the django-rest framework and the flask-API framework. According to the netsolutions.com website, a framework can be described as a tool that provides ready made components or solutions that are customized in order to speed up software development. Flask-API framework is a smaller framework as compared to python’s Django. Both Flask and Django-REST framework make good but the difference lies in the efficiency in development by reducing the time for development and the effective functioning of the API. According to the hackr.io website Django, on the one hand, is a full-stack web framework, whereas Flask is a light-weight, extensible framework. If you want to dig more into coding and learn core concepts, Flask helps you understand how each component from the back-end works to get a simple web application up and running. Django follows lots of design patterns, and hence you learn a lot of exciting concepts. Further, with both frameworks, there is a whole lot of community support and documentation, so if you ever run into any problems, the chances are that the problem has been already discussed and sorted out – something that makes your work easier. Django is heavy-weight, whereas, with Flask, you have to build everything on your own. According to the same website, Django has been in existence from around 2005; while Flask came in around 2010 – that is 5 years later after django. However, these two are both top frameworks in the Python world in current times. Their growing popularity is evident, considering the number of questions people ask about both of these frameworks in various tech forums and tech websites. I decided to deploy my model using Django as it has ready built components as opposed to flask which requires one to start the job from the ground up; this attribute of Django makes it efficient for fast development. Django provides tools for building REST APIs. According to the redhat.com website, a REST API (also known as a RESTFUL API ) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding. I therefore created my API using the Django-rest framework as the best option in my case to deploy the trained model.

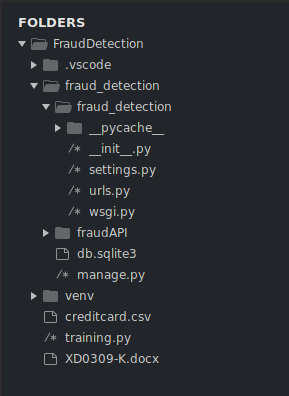
# **BUILDING A REST API TO DEPLOY THE MODEL**

# To build the model I needed to have the Django framework installed on my computer as it does not come with the standard python library. It is considered a best practice in python(and probably in other programming languages) to install all the needed libraries within a virtual environment. According to the geekforgeeks.org website, a virtual environment is a tool that helps to keep dependencies required by other projects seperate by creating an isolated python virtual directories for them. This ensures that the project is portable and can be run on any computers without the need to reinstall the required libraries. This is mostly important during the development phase of the software. I created my project in a new directory that I named ‘FraudDetection’. After creating the project I created a virtual environment using the python ‘virtualenv’ command. The syntax used is: ‘virtualenv VirtualEnvironmentName’. To create a virtual environment named ‘venv’(which is the short form of – virtual environment) one would use the command ‘virtualenv venv’. It is also important to note that the command to create a virtual environment won’t work unless the library exists within the python program installed within the computer. To install the virtualenv library the following command is used; ‘pip install virtualenv’. It is also important to note that an outdated version of the virtualenv library may fail to work as required. The python pip (According to w3schools.com, pip is a python package manager for python used in the installation and upgrading of various python third party packages)install command installs the latest version of any software by default but in a case where the installed pip is outdated, one can run the pip install virtualenv command to install the latest version. After the creation of a virtual environment one can activate the virtual environment by changing the current directory to the virtual environment’s directory and running the source bin/activate command. This can be done as shown below.

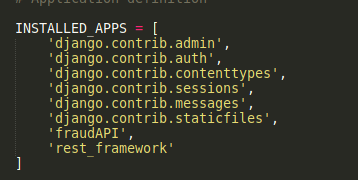
* 

# With the virtual environment created we can proceed to install the required packages by using the following pip commands: pip install django, django\_rest\_framework. This can be done as shown below.

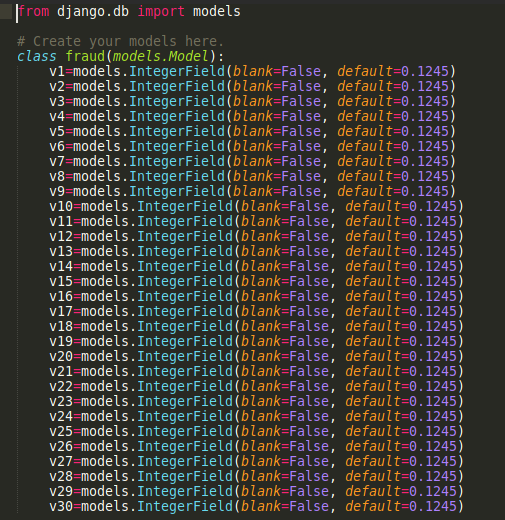
# Having installed the needed dependencies, I can now proceed to create my Django project using the ‘django-admin startproject projectname’ command. This creates a project folder with the following files; settings.py, manage.py, urls.py, \_\_init\_\_.py, wsgi.py and a dbsqlite3 database file. After creating the project one should change directories to the project folder and run the Django ‘python manage.py startapp appname’ command to create an app folder within the project folder. The folder created contains the following python files; views.py file – this file contains all the logic, models.py file – this file contains classes that represent database tables and is Django’s way of providing a simpler, easier and efficient way of creating a database, tests.py file – this file contains tests written to manage any errors that may occur due to logic, apps.py file – which contains an interface for registering the app to the project, the admin.py files – which is a way through which database tables are registered to the admin interface. It is in this folder where one can add any extra files. After creating the project and app folders one should end up with the following directory structure.

****

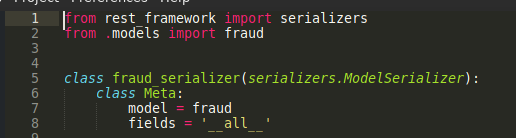
# After every package installation and creation of a new app folder within a project it’s important the installed packages to the installed\_apps list within the project settings.py file. This helps to inform the project of the existing apps and packages. For that reason the installed django\_rest\_framework and created app(the app’s name is fraudAPI) is added to the installed apps list as shown below.

****

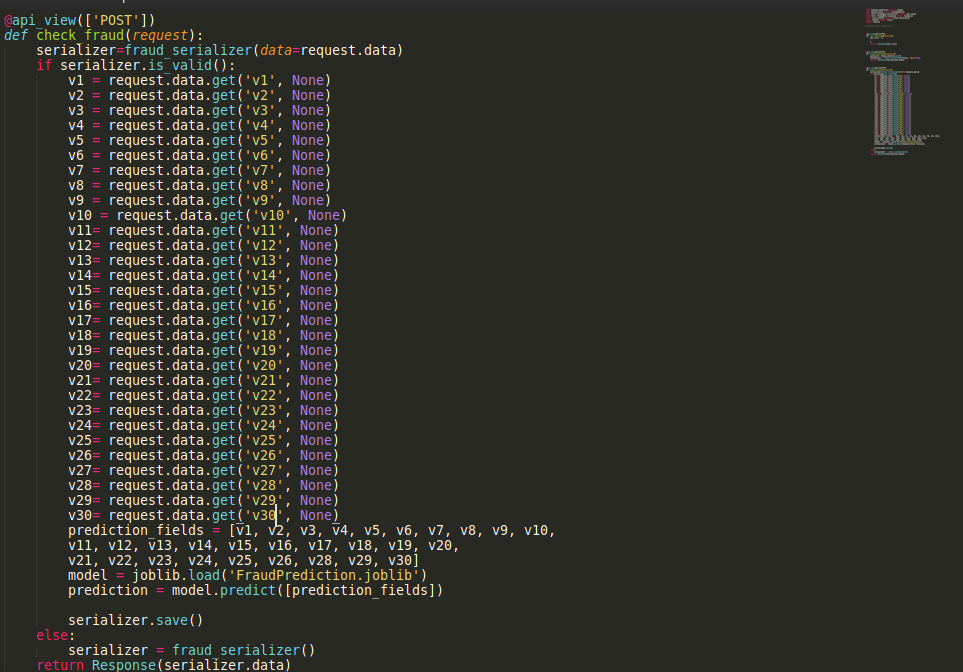
# After the installation I created a database using the Django models class. Django creates databases by first creating an instance of the Django.models class. The class is then given attributes that much a tables columns. This fastens the rate at which a database is created. To create a database that would suit my needs I created a model with a number of attributes equal to the number of parameters needed by the model to make a prediction. The model can be created as shown below.

* 

# This creates a model with 30 attributes of type integer which is accessed by extending the models.IntegerField() class. This corresponds to the integer type of databases and means that only a value of type integer can be stored within that field of the database. To make changes to the database file within the project folder the following Django commands are run; python manage.py makemigrations – to create migrations for the saved changes to the database model and python manage.py migrate to make the changes to the database. After creation of the database, I created a serializers.py file which contained my serializers for the API. Serializers in Django can be thought of as forms that are used to make changes to the database by either creating, deleting or updating.

* I needed to create serializers with post methods to enable retrieval of data sent by the various apps that would be later passed to the model for a prediction before being saved to the database. The serializers can be created by the code shown below.
* 

After creating the serializer I wrote the logic in the views.py file for passing the retrieved data to our model( it is also important to move the file containing the model into a folder where it can be accessed by the views.py file; most probably the same folder as the views.py file).

The code involves taking in input from the applications and appends it to a list before passing the list to the model for prediction and storing it in a variable for later access; the code is as shown below.

**CONCLUSION**

Having carried out the following research and terminated the project I came to the following realizations as will be outlined in the rest of this section.

Machine learning is a potent tool in the hands of the right developer and with the best practices can be used quite effectively. It is also evident that there may be a possibility to develop a standard procedure for detecting fraudulent transactions through use of standard parameters; the problem however arises from the fact that different transaction systems use various parameters for transaction. To develop a standard set of parameters one only has to consider the values that appear constant across multiple transaction processing systems.

In conclusion it will be worth noting that fraud detection through machine learning is highly possible and given time there will be a standard set of procedures through which fraudulent transactions can be detected. It is also worth noting the cost reduction that will be incurred by employing machine learning procedures over conventional methods involving manual labor by human beings and other methods such as two step authentication.

# **References**

Bhanusri, A., Valli, K. R. S., Jyothi, P., Sai, G. V., & Subash, R. R. S. (2020). Credit card fraud detection using Machine learning algorithms. Journal of Research in Humanities and Social Science, 8(2), 04-11.

Goyal, R., & Manjhvar, A. K. (2020). Review on Credit Card Fraud Detection using Data Mining Classification Techniques & Machine Learning Algorithms. IJRAR-International Journal of Research and Analytical Reviews (IJRAR), E-ISSN, 2348-1269.

Borah, L., Saleena, B., & Prakash, B. CREDIT CARD FRAUD DETECTION USING DATA MINING TECHNIQUES. Journal of Seybold Report ISSN NO, 1533, 9211.

Abdou, A. E. H. A., Mohammed, H. E., Khalifa, W., Roushdy, M. I., & Salem, A. B. M. (2019). Machine Learning Techniques for Credit Card Fraud Detection. Future Computing and Informatics Journal, 4(2), 5.

Eddison, L(2018). Machine Learning Model. Python Machine Learning. Duman, E., & Ozcelik, M. H. J. E. S. w. A. (2011). Detecting credit card fraud by genetic algorithm and scatter search. 38(10), 13057-13063.

Excell, D. J. C. F., & Security. (2012). Bayesian inference–the future of online fraud protection. 2012(2), 8-11.

Foster, D. P., & Stine, R. A. J. J. o. t. A. S. A. (2004). Variable selection in data mining: Building a predictive model for bankruptcy. 99(466), 303-313.

Gaikwad, J. R., Deshmane, A. B., Somavanshi, H. V., Patil, S. V., Badgujar, R. A. J. I. J. o. I. T., & Engineering, E. (2014). Credit Card Fraud Detection using Decision Tree Induction Algorithm. 4(6).

Ghosh, S., & Reilly, D. L. (1994). Credit card fraud detection with a neural-network. Paper presented at the System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on.

Juszczak, P., Adams, N. M., Hand, D. J., Whitrow, C., Weston, D. J. J. C. S., & Analysis, D. (2008). Off-the-peg and bespoke classifiers for fraud detection. 52(9), 4521-4532.

Khan, A. U. S., Akhtar, N., & Qureshi, M. N. (2014). Real-time credit-card fraud detection using artificial neural network tuned by simulated annealing algorithm. Paper presented at the Proceedings of International Conference on Recent Trends in Information, Telecommunication and Computing, ITC.

Kulkarni, P., & Ade, R. (2016). Logistic regression learning model for handling concept drift with unbalanced data in credit card fraud detection system. Paper presented at the Proceedings of the Second International Conference on Computer and Communication Technologies.

Lakshmi Narayana S., Suneetha Devi J., Bhargav Reddy I., Harish Paruchuri. (2012). Optimizing Voice Recognition using Various Techniques. CiiT International Journal of Digital Signal Processing, 4(4), 135-141

Lu, Q., & Ju, C. J. J. o. C. I. T. (2011). Research on credit card fraud detection model based on class weighted support vector machine. 6(1).

Maes, S., Tuyls, K., Vanschoenwinkel, B., & Manderick, B. (2002). Credit card fraud detection using Bayesian and neural networks. Paper presented at the Proceedings of the 1st international naiso congress on neuro fuzzy technologies.

Masuda, B. J. C. p. s. (1993). Credit card fraud prevention: A successful retail strategy. 1, 121-134.

Movva, L., Kurra, C., Koteswara Rao, G., Battula, R. B., Sridhar, M., & Harish, P. (2012). Underwater Acoustic Sensor Networks: A Survey on MAC and Routing Protocols. International Journal of Computer Technology and Applications, 3(3).

Murli, D., Jami, S., Jog, D., Nath, S. J. I. J. o. S. R. i. T., & Management. (2015). Credit card fraud detection using neural networks. 2(2), 84-88.

Ngai, E. W., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. J. D. s. s. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. 50(3), 559-569.

Ogwueleka, F. N. J. J. o. E. S., & Technology. (2011). Data mining application in credit card fraud detection system. 6(3), 311-322.

Özçelik, M. H., Duman, E., Işik, M., & Çevik, T. (2010). Improving a credit card fraud detection system using genetic algorithm. Paper presented at the 2010 International Conference on Networking and Information Technology.