**BITCOIN PRICE PREDICTION USING GRU ALGORITH**

**Abstract:**

Blockchain technology is becoming increasingly popular because of its applications in various fields. It gives an edge over the traditional centralized methods as it provides decentralization, immutability, integrity, and anonymity. The most popular application of this technology is bitcoin , which showed a massive rise in their popularity and market capitalization in recent years. Individual investors, big institutions, and corporate firms are investing heavily in it. However, the crypto market is less stable than traditional commodity markets. It can be affected by many technical, sentimental, and legal factors, so it is highly volatile, uncertain, and unpredictable. Plenty of research has been done on various bitcoin to forecast accurate prices, but the majority of these approaches cannot be applied in real-time. Motivated from the aforementioned discussion, in this paper, we propose a deep-learning-based hybrid model (includes Gated Recurrent Units (GRU)) to predict the price of Litecoin and Zcash with inter-dependency of the parent coin. The proposed model can be used in real-time scenarios and it is well trained and evaluated using standard data sets. Results illustrate that the proposed model forecasts the prices with high accuracy compared to existing models.

**Introduction:**

Modern monetary systems are based on fiat money, which has many advantages because of its divisibility, transfer-ability, durability, and scarcity [1]. However, there are a couple of problems with it such as the currency is not backed by anything, so governments have control over money. It can lead to many issues, such as hyperinflation and income inequality [2]. Yugoslavia, Peru, and Venezuela are suffering from hyperinflation because the current system has failed [3]. The second problem with the current system is the vulnerability of existing ledgers, which keep the record of all transactions. The modern financial system says that money is just an entry on these ledgers, but they can be manipulated and violated. The third problem is the way people transact money. Everyone transacts money with a cheque, wire transfers, credit cards, or online applications such as G-pay or Amazon Pay, etc. The payment goes through a financial institution or intermediaries such as credit card companies, clearinghouses, and financial institutions. The average cost of transferring money from one country to another ranges from 6% to 10% and can sometimes take up to one week to complete the transaction. People have lost control and ownership of their data because of the monopoly of these intermediaries. People trust these institutions because of their accountability and predictability. Based on trust, more than 6 billion people transact 200 trillion worth of money every year [4]. This trust is backed by the government regulations and legal contracts, but it is easily breakable and the world has witnessed many instances of trust breach, such as the crash of the dot-com bubble in the 1990s and the real-estate bubble in 2008, which had wiped out trillions of dollars [5]. A question arises how people trust the current financial system, which has a threat of hyperinflation, the ledgers are not tamper proof, and the intermediaries can also be failed. So there is a need to develop a model, which establishes the trust among all the stakeholders. On 31 October 2008, Satoshi Nakamoto proposed a system that revolutionized the current system with the invention of a technology called Blockchain having first digital currency Bitcoin [6]. Bitcoin is a peer-to-peer (P2P) money transferring system that allows users to transact digital money over the public internet without intermediaries [7]. Blockchain stores all the transactions in the forms of blocks and all the blocks have their unique key. A block contains cryptographically encoded data locked by its key and the data of the previous block. In this manner, it creates the whole chain of blocks. It becomes nearly impossible to tamper any existing record or to get the control of this ledger, so it is a secure, immutable, and tamper proof ledger running on a decentralized network of computers. The integrity of the system is maintained by all the users through consensus algorithms, public-key cryptography methods, smart contracts, hashes, and digital handshakes [8]. Using bitcoin people can transfer money anytime at minimal cost instantly. Also, this technology can help to reduce inflation and income inequality. It has the potential to close the ever-increasing trust gap between investors and sellers. It can solve the double-spending problem and detect fraud and users can achieve true data democracy with the help of it [9], [10]. Many bitcoin entered into the crypto market after Bitcoin, for example, Ethereum, launched in 2015, is the second-largest bitcoin with a $410 billion market capitalization [11]. More than 5,600 different bitcoin are traded in around 1,100 exchanges and Ripple, Tether, Cardano, Stellar, Litecoin, and Zcash are the most popular digital currencies. Back in June 2016, the total market capitalization of all bitcoin was approximately 12.22 billion dollars and it fluctuated in 2017. It increased to $1.75 trillion in June 2021 [12], with an all-time high of $2 trillion. It will reach nearly $8 trillion by 2030. The daily volume of the crypto market is around $117 billion and more than 100 million people are using these currencies [13]. The Crypto market is attracting more people with its high returns and rapid growth. bitcoin have become an intangible digital assets for many individual investors and traders to invest in them [14]. It is a new investment opportunity for financial institutions, hedge funds, and corporate companies [15]. This market can grow exponentially and its growth is also evident after August 2020 during the COVID-19 pandemic [16]. Also, the researchers, companies, start-ups, and universities across the globe are working to make this new technology more reliable, mature, and secure Many researchers are working to make crypto mining more efficient, cheaper [17], [18] and to prevent cryptojacking [19]. Predicting the accurate price of bitcoin is always challenging because of their volatility and complexity. The price of crypto depends on more than 25 technical factors and market sentiment. Many bitcoin, such as Litecoin and Zcash, are dependent on major c bitcoin, such as Bitcoin. Moreover, government and international regulations and plenty of legal factors can affect their prices [20]–[22]. Because of all these factors, many bitcoin have shown more than 30% of growth in a single day, which is highly unpredictable and unreliable for the investors. However, with the use of various Machine Learning (ML) and Deep Learning (DL) algorithms, forecasting has become a little bit easier than past [23]. Plenty of research has been done in this area and many investors and financial institutions are trading with their price prediction system [24]. Motivated from the aforementioned discussions, in this paper, we propose a deep learning-based scheme to predict the accurate price of Litecoin and Zcash. We have trained the proposed model with the data of the last five years, tested it in the realtime, and compared its results with actual prices.

**Literature Survey:**

**Title:** Monetary Discipline, Germany, and the European Monetary System

**Author:** Jacques Melitz

**Year:** February 17, 1987

**Abstract:** This paper explores the hypothesis that the non-German members of the European Monetary System (EMS) draw benefits from the system because of the monetary discipline that it imposes upon them. The hypothesis explains the dominant position of Germany in the EMS and is consistent with the evidence that membership has induced several countries to disinflate more than they would have done otherwise. Analysis shows, however, that the required conditions for the hypothesis to work are very stringent. Even if the conditions are met, the non-German members of the EMS could obtain the advantages of monetary discipline in other ways.

**Title:** Cryptocurrency Trading Using Machine Learning

**Author:** Thomas E. Koker's

**Year:** August 2020

**Abstract:** We present a model for active trading based on reinforcement machine learning and apply this to five major cryptocurrencies in circulation. In relation to a buy-and-hold approach, we demonstrate how this model yields enhanced risk-adjusted returns and serves to reduce downside risk. These findings hold when accounting for actual transaction costs. We conclude that real-world portfolio management application of the model is viable, yet, performance can vary based on how it is calibrated in test samples

**Title**: Bitcoin Price Forecasting using LSTM and 10-Fold Cross validation

**Author:** Sakshi Tandon's

**Year:** March 2019

**Abstract:** —This research paper reports the proposed model for price prediction of the popular Bitcoin crypto currency while applying different neural network approaches namely Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM) along with 10-fold cross validation. In this work, the analysis of various trends of Bitcoin market is carried out and learning of important features used for price prediction is done. Daily price change is estimated by the neural network models. New activation functions are utilized in this research paper for improving efficiency. Further, this research paper compares the proposed model with other existing models namely; RNN with LSTM, Linear Regression and Random Forest applied in the same domain. The dataset utilized in this work is taken from the website named coinmarket and live streaming data is considered for the experimental work. Keras, Tensorflow and Scikit Learn have been used for performing the experimental work of the proposed model. The performance analysis of the proposed model with the existing ones has been carried out in terms of the Mean Absolute Error (MAE). It is observed from the results retrieved as a part of this work that the MAE for the proposed model came out to be 0.0043s which was significantly less than its existing counterparts

**Existing System:**

In existing system we analyzed stock markets prediction, suggests that these methods could be effective also in predicting crypto currencies prices. However, the application of machine learning algorithms to the crypto currency market has been limited so far to the analysis of Bitcoin prices, using random forests, Bayesian neural network, long short-term memory neural network  and other algorithms. These studies were able to anticipate, to different degrees, the price fluctuations of Bitcoin, and revealed that best results were achieved by neural network based algorithms. Deep reinforcement learning was showed to beat the uniform buy and hold strategy in predicting the prices of 12 crypto currencies over one year period.

**Disadvantage:**

1. Other attempts to use machine learning to predict the prices of cryptocurrencies other than Bitcoin come from non-academic sources.
2. Most of these analyses focused on a limited number of currencies and did not provide benchmark comparisons for their results.

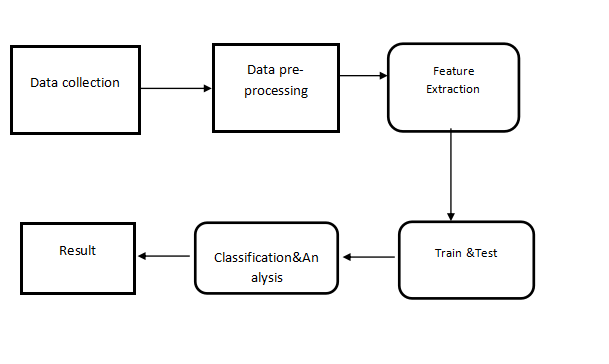
**Proposed System:**

Here, we test the performance of three models in predicting daily cryptocurrency price for 1,681 currencies. Two of the models are based on gradient boosting decision trees and one is based on **Gated Recurrent Units**  (GRU). In all cases, we build investment portfolios based on the predictions and we compare their performance in terms of return on investment. We find that all of the three models perform better than a baseline ‘simple moving average’ model  where a currency’s price is predicted as the average price across the preceding days, and that the method based on long short-term memory recurrent neural networks systematically yields the best return on investment.

**Advantage:**

1. We present and compare the results obtained with the three forecasting algorithms and the baseline method.
2. The analysis considers all currencies whose age is larger than 50 days since their first appearance and whose volume is larger than $100000.
3. To discount for the effect of the overall market movement (i.e., market growth, for most of the considered period), we consider crypto currencies prices expressed in Bitcoin.
4. Increasing the highest accuracy.
5. Time consumption is very less.

**System Architecture:**



**System Requirements:-**

**Software and Hardware Requirements:**

**Hardware:**

 OS – Windows 7, 8 and 10 (32 and 64 bit)

RAM – 4GB

**Software:**

Python / Anaconda Navigator

**Feasibility study**

Feasibility study in the sense it's a practical approach of implementing the proposed model of system . Here for a machine learning projects .we generally collect the input from online websites and filter the input data and visualize them in graphical format and then the data is divided for training and testing . That training is testing data is given to the algorithms to predict the data .

1. First, we take dataset.

2. Filter dataset according to requirements and create a new dataset which has attribute according to analysis to be done

3. Perform Pre-Processing on the dataset

4. Split the data into training and testing

5. Train the model with training data then analyze testing dataset over classification algorithm

6. Finally you will get results as accuracy metrics.

**Modules**

1. **DATA COLLECTION**
2. **DATA PRE-PROCESSING**
3. **FEATURE EXTRATION**
4. **EVALUATION MODE**

**DATA COLLECTION**

Data collection is a process in which information is gathered from many sources which is later used to develop the machine learning models. The data should be stored in a way that makes sense for problem. In this step the data set is converted into the understandable format which can be fed into machine learning models.

Data used in this paper is a set of cervical cancer data with 15 features . This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called *labelled data*.

**DATA PRE-PROCESSING**

Organize your selected data by formatting, cleaning and sampling from it.

Three common data pre-processing steps are:

**Formatting**: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.

**Cleaning:** Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

**Sampling:** There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

**FEATURE EXTRATION**

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes.  Finally, our models are trained using Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data. The chosen classifiers were Random forest. These algorithms are very popular in text classification tasks.

**EVALUATION MODEL**

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over fitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid over fitting, both methods use a test set (not seen by the model) to evaluate model performance.

Performance of each classification model is estimated base on its averaged. The result will be in the visualized form. Representation of classified data in the form of graphs.

**Accuracy** is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

**DATA FLOW DIAGRAM**

**LEVEL 0**

**LEVEL 1**

**LEVEL 2**

**Algorithms used:**

**GRU(Gated Recurrent Units):**

Gated Recurrent Units (GRUs) are a gating method in Recurrent Neural Networks, first proposed by Kyunghyun Cho et al. in 2014. GRUs are improved version of LSTM Networks, but they are so special and Effective than the LSTM Networks. The goal of the GRU is to solve the vanishing gradient problem that is encountered by a standard Recurrent Neural Network. GRU uses 2 gates to solve the vanishing gradient problem, they are Update gate and Reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.

In this system, the data set is taken from yahoo finance. The data set taken as70 to 30 ratio for training and testing ratio. The data set contains Close, open, date, volume. The dataset having dates of interval a day i.e., day by day. It also performs well in multi-class prediction. By using this model, the accuracy and performance of prediction of Bitcoin price are increased.**Front end(web application):**

**STREAMLIT:**

Streamlit is an open-source (free) Python library, which provides a fast way to build interactive web apps. It is a relatively new package but has been growing tremendously. It is designed and built especially for machine learning engineers or other data science professionals. Once you are done with your analysis in Python, you can quickly turn those scripts into web apps/tools to share with others.

As long as you can code in Python, it should be straightforward for you to write Streamlit apps. Imagine the app as the canvas. We can write Python scripts from top to bottom to add all kinds of elements, including text, charts, widgets, tables, etc.

Streamlit is the easiest way especially for people with no front-end knowledge to put their code into a web application:

 No front-end (html, js, css) experience or knowledge is required.

 You don't need to spend days or months to create a web app, you can create a really beautiful machine learning or data science app in only a few hours or even minutes.

 It is compatible with the majority of Python libraries (e.g. pandas, matplotlib, seaborn, plotly, Keras, PyTorch, SymPy(latex)).

 Less code is needed to create amazing web apps.

 Data caching simplifies and speeds up computation pipelines.

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.dates as mdates

from keras.models import Sequential

from keras.layers import Dense, Dropout,GRU

from keras import optimizers

from sklearn.preprocessing import MinMaxScaler

seed = 1234

np.random.seed(seed)

plt.style.use('ggplot')

import warnings

warnings.filterwarnings("ignore")

# Import Data

bitcoin = pd.read\_csv('BTC-USD.csv',index\_col='Date', parse\_dates=['Date'])

bitcoin

bitcoin.head()

bitcoin.tail()

bitcoin.shape

bitcoin.info()

bitcoin.isnull().sum()

bitcoin.duplicated().sum()

bitcoin.isna().sum()

df = pd.DataFrame(bitcoin['Close'])

print(' Count row of data: ',len(df))

df.describe()

dataset\_norm = df.copy()

df[['Close']]

scaler = MinMaxScaler()

dataset\_norm['Close'] = scaler.fit\_transform(df[['Close']])

dataset\_norm

fig = plt.figure(figsize=(10, 4))

plt.plot(dataset\_norm)

plt.xlabel('Date')

plt.gca().xaxis.set\_major\_formatter(mdates.DateFormatter("%Y-%m"))

plt.title('Data Normalized')

plt.show()

# Partition data into data train, val & test

totaldata = df.values

totaldatatrain = int(len(totaldata)\*0.7)

totaldataval = int(len(totaldata)\*0.1)

totaldatatest = int(len(totaldata)\*0.2)

# Store data into each partition

training\_set = dataset\_norm[0:totaldatatrain]

val\_set=dataset\_norm[totaldatatrain:totaldatatrain+totaldataval]

test\_set = dataset\_norm[totaldatatrain+totaldataval:]

# graph data training

fig = plt.figure(figsize=(10, 4))

plt.plot(training\_set)

plt.xlabel('Date')

plt.gca().xaxis.set\_major\_formatter(mdates.DateFormatter("%Y-%m"))

plt.title('Data Training')

plt.show()

fig = plt.figure(figsize=(10, 4))

plt.plot(val\_set)

plt.xlabel('Date')

plt.gca().xaxis.set\_major\_formatter(mdates.DateFormatter("%Y-%m"))

plt.title('Data Validation')

plt.show()

fig = plt.figure(figsize=(10, 4))

plt.plot(test\_set)

plt.xlabel('Date')

plt.gca().xaxis.set\_major\_formatter(mdates.DateFormatter("%Y-%m"))

plt.title('Data Test')

plt.show()

lag = 2

def create\_sliding\_windows(data,len\_data,lag):

x=[]

y=[]

for i in range(lag,len\_data):

x.append(data[i-lag:i,0])

y.append(data[i,0])

return np.array(x),np.array(y)

# Formating data into array for create sliding windows

array\_training\_set = np.array(training\_set)

array\_val\_set = np.array(val\_set)

array\_test\_set = np.array(test\_set)

# Create sliding windows into training data

x\_train, y\_train = create\_sliding\_windows(array\_training\_set,len(array\_training\_set), lag)

x\_train = np.reshape(x\_train, (x\_train.shape[0],x\_train.shape[1],1))

# Create sliding windows into validation data

x\_val,y\_val = create\_sliding\_windows(array\_val\_set,len(array\_val\_set),lag)

x\_val = np.reshape(x\_val, (x\_val.shape[0],x\_val.shape[1],1))

# Create sliding windows into test data

x\_test,y\_test = create\_sliding\_windows(array\_test\_set,len(array\_test\_set),lag)

x\_test = np.reshape(x\_test, (x\_test.shape[0],x\_test.shape[1],1))

x\_val

y\_val

plt.figure(figsize=(8, 4), dpi=80)

plt.hist(df.Close, bins=20, color='k')

plt.title('Close')

plt.show()

plt.figure(figsize=(8, 4), dpi=80)

plt.hist(bitcoin.High, bins=20, color='r')

plt.title('Adj Close of the bitcoin')

plt.show()

# Model GRU

# Hyperparameters

learning\_rate = 0.0001

hidden\_unit = 64

batch\_size=256

epoch = 100

# Architecture Gated Recurrent Unit

regressorGRU = Sequential()

# First GRU layer with dropout

regressorGRU.add(GRU(units=hidden\_unit, return\_sequences=True, input\_shape=(x\_train.shape[1],1), activation = 'tanh'))

regressorGRU.add(Dropout(0.2))

# Second GRU layer with dropout

regressorGRU.add(GRU(units=hidden\_unit, return\_sequences=True, activation = 'tanh'))

regressorGRU.add(Dropout(0.2))

# Third GRU layer with dropout

regressorGRU.add(GRU(units=hidden\_unit, return\_sequences=False, activation = 'tanh'))

regressorGRU.add(Dropout(0.2))

# Output layer

regressorGRU.add(Dense(units=1))

# Compiling the Gated Recurrent Unit

regressorGRU.compile(optimizer=optimizers.Adam(lr=learning\_rate),loss='mean\_squared\_error')

# Fitting ke data training dan data validation

pred = regressorGRU.fit(x\_train, y\_train, validation\_data=(x\_val,y\_val), batch\_size=batch\_size, epochs=epoch)

# Graph model loss (train loss & val loss)

fig = plt.figure(figsize=(10, 4))

plt.plot(pred.history['loss'], label='train loss')

plt.plot(pred.history['val\_loss'], label='val loss')

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(loc='upper right')

plt.show()

# Tabel value of training loss & validation loss

learningrate\_parameter = learning\_rate

train\_loss=pred.history['loss'][-1]

validation\_loss=pred.history['val\_loss'][-1]

learningrate\_parameter=pd.DataFrame(data=[[learningrate\_parameter, train\_loss, validation\_loss]],

columns=['Learning Rate', 'Training Loss', 'Validation Loss'])

learningrate\_parameter.set\_index('Learning Rate')

# Implementation model into data test

y\_pred\_test = regressorGRU.predict(x\_test)

# Invert normalization min-max

y\_pred\_invert\_norm = scaler.inverse\_transform(y\_pred\_test)

# Comparison data test with data prediction

datacompare = pd.DataFrame()

df=np.array(df['Close'][totaldatatrain+totaldataval+lag:])

datapred= y\_pred\_invert\_norm

datacompare['Data Test'] = df

datacompare['Prediction Results'] = datapred

datacompare

# Calculatre value of Root Mean Square Error

from sklearn.metrics import r2\_score

def rmse(df, datapred):

return np.round(np.sqrt(np.mean((datapred - df) \*\* 2)), 4)

print('Root Mean Square Error :',rmse(df, datapred))

def mape(datatest, datapred):

return np.round(np.mean(np.abs((df - datapred) / df) \* 100), 4)

print('Mean Absolute Percentage Error : ', mape(df, datapred), '%')

print(r2\_score(df, datapred))

# Create graph data test and prediction result

plt.figure(num=None, figsize=(10, 4), dpi=80,facecolor='w', edgecolor='k')

plt.title('Graph Comparison Data Actual and Data Prediction')

plt.plot(datacompare['Data Test'], color='red',label='Data Test')

plt.plot(datacompare['Prediction Results'], color='blue',label='Prediction Results')

plt.xlabel('Day')

plt.ylabel('Price')

plt.legend()

plt.show()

import pickle

filename='bitcoin.sav'

pickle.dump(regressorGRU ,open('bitcoin.sav','wb'))

loaded\_model=pickle.load(open('bitcoin.sav','rb'))

**UML Diagrams**

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

* actors
* business processes
* (logical) components
* activities
* programming language statements
* database schemas, and
* Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

**Sequence Diagram:**

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

**Activity Diagrams-:**

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency.In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**Usecase diagram:**

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously putting effort to make a truly industry standard.

UML stands for Unified Modeling Language.

UML is a pictorial language used to make software blue prints

**Class diagram**

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

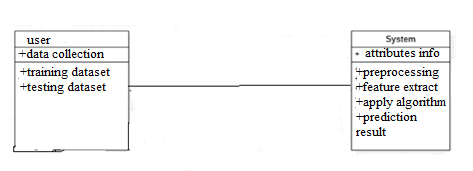
In the diagram, classes are represented with boxes that contain three compartments:

The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.

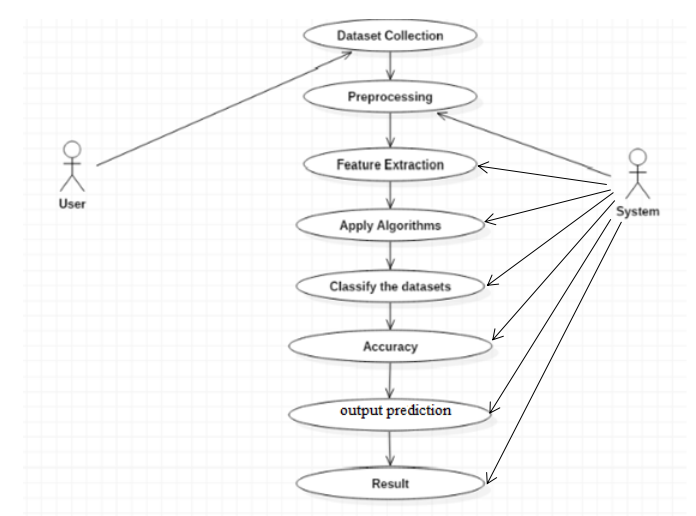
The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.

The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

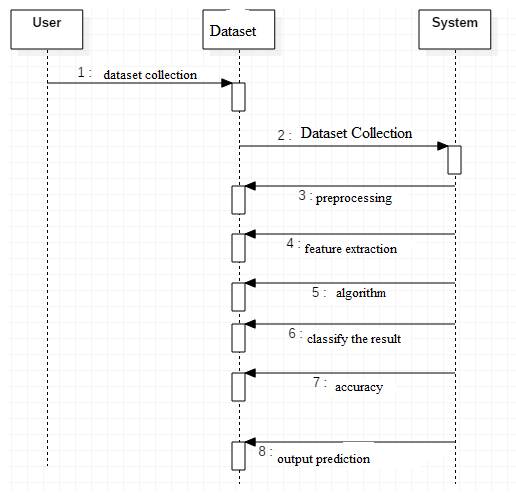
**Use Case Diagram**



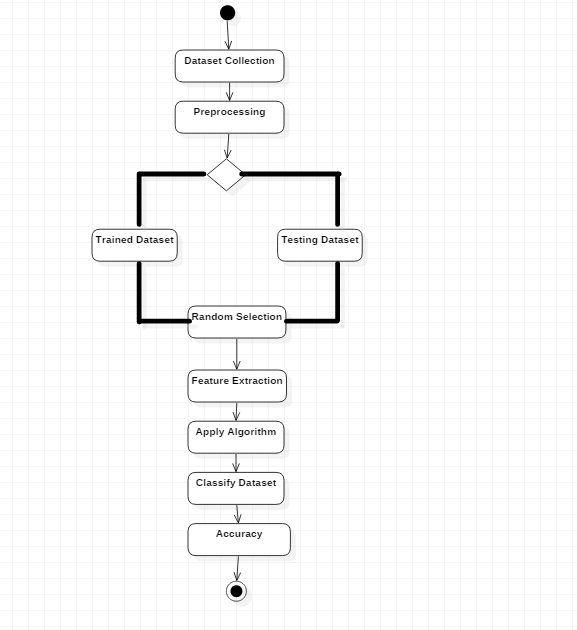
**CLASS DIAGRAM**



SEQUENCE DIAGRAM



**ACTIVITY DIAGRAM**



**PYTHON OVERVIEW**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted:** Python is processed at runtime by the interpreter.You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive:** You can actually sit at a Python prompt and interactwith the interpreter directly to write your programs.
* **Python is Object-Oriented:** Python supports Object-Oriented style ortechnique of programming that encapsulates code within objects.
* **Python is a Beginner's Language:** Python is a great language for thebeginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include:

**Easy-to-learn:** Python has few keywords, simple structure, and a clearlydefined syntax. This allows the student to pick up the language quickly.

**Easy-to-read:** Python code is more clearly defined and visible to the eyes.

**Easy-to-maintain:** Python's source code is fairly easy-to-maintain.

**A broad standard library:** Python's bulk of the library is very portable andcross-platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode:** Python has support for an interactive mode which allowsinteractive testing and debugging of snippets of code.

**Portable:** Python can run on a wide variety of hardware platforms and has thesame interface on all platforms.

**Extendable:** You can add low-level modules to the Python interpreter. Thesemodules enable programmers to add to or customize their tools to be more efficient.

**Databases:** Python provides interfaces to all major commercial databases.

**GUI Programming:** Python supports GUI applications that can be created andported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable:** Python provides a better structure and support for large programsthan shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

* IT supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

**ANACONDA NAVIGATOR**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

## Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

## **WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR**?

The following applications are available by default in Navigator:

* Jupyter Lab
* Jupyter Notebook
* QT Console
* Spyder
* VS Code
* Glue viz
* Orange 3 App
* Rodeo
* RStudio

Advanced conda users can also build your own Navigator applications

## How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

## What’s new in 1.9?

* Add support for **Offline Mode** for all environment related actions.
* Add support for custom configuration of main windows links.
* Numerous bug fixes and performance enhancements.

**TESTING**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software Testing can also be stated as the process of validating and verifying that a software program/application/product:

* Meets the business and technical requirements that guided its design and Development.
* Works as expected and can be implemented with the same characteristics.

**TESTING METHODS**

* **Functional Testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

* Functions: Identified functions must be exercised.
* Output: Identified classes of software outputs must be exercised.
* Systems/Procedures: system should work properly

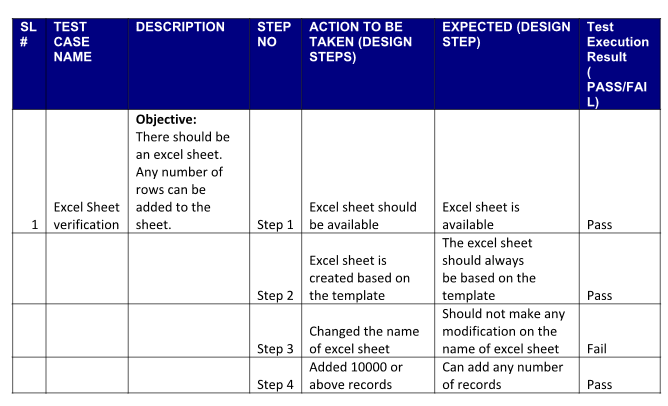
**Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

Test Case for Excel Sheet Verification:

Here in machine learning we are dealing with dataset which is in excel sheet format so if any test case we need means we need to check excel file. Later on classification will work on the respective columns of dataset .

Test Case 1 :



**MODEL IMPLEMENTATION AND RESULTS :**

The Data set is taken from yahoo finance, and it contains 7 columns namely date, close, open, low, high, adj close and volume. Where open and close are opening and closing price of the bitcoin and the prices are in dollars. The values are taken per day. The data values are normalized and divided into training set and test set. The actual and predicted values of GRU .The MAE and RMSE of the two models ,GRU model with recurrent dropout predicted Bitcoin price in the test data, as compared to the original data. The model predicted price is close to the original price compared with gru

**CONCLUSION**

The performance of the model will increase when compared to previous individual models. We find that the GRUs models provide more accurate predictions for the bitcoin market compared to the other models. As RNN uses less memory, the training and execution will be faster. The RMSE losses (Root Mean Squared Error) are reduced when compared to the individual models. The MAE losses (Mean Absolute Error) are reduced when compared to the individual models.

**REFERENCES**

**1.**B. Busman, N. Nurhayati, F. Amali and Z. Muttaqin, "Penerapan Big Data Pada Forex Trading Menggunakan Analisa Statistik Dengan Breakout Strategy", Pseudocode, vol. 4, no. 2, pp. 137-143, 2017.

**2.**H. R. Makridakis, S. and S.C. Wheelwright, "1 / the Forecasting Perspective", Forecast. methods Appl., pp. 1-632, 1997.

**3.**S. M. Idrees, M. A. Alam and P. Agarwal, "A Prediction Approach for Stock Market Volatility Based on Time Series Data", IEEE Access, vol. 7, no. c, pp. 17287-17298, 2019.

**4.**B. Nakisa, M. N. Rastgoo, A. Rakotonirainy, F. Maire and V. Chandran, "Long short term memory hyperparameter optimization for a neural network based emotion recognition framework", IEEE Access, vol. 6, no. c, pp. 49325-49338, 2018.

**5.**Busman Nurhayati and V. Amrizal, "Big Data Analysis Using Hadoop Framework and Machine Learning as Decision Support System (DSS) (Case Study: Knowledge of Islam Mindset)", 2018 6th Int. Conf. Cyber IT Serv. Manag. CITSM 2018, pp. 1-6, 2019.

**6.**W. Yiying and Z. Yeze, "Cryptocurrency Price Analysis with Artificial Intelligence", 5th Int. Conf. Inf. Manag. ICIM 2019, pp. 97-101, 2019.

**7.**T. Phaladisailoed and T. Numnonda, "Machine learning models comparison for bitcoin price prediction", Proc. 2018 10th Int. Conf. Inf. Technol. Electr. Eng. Smart Technol. Better Soc. ICITEE 2018, pp. 506-511, 2018.

**8.**A. Shewalkar, D. Nyavanandi and S. A. Ludwig, "Performance Evaluation of Deep neural networks Applied to Speech Recognition: Rnn LSTM and GRU", J. Artif. Intell. Soft Comput. Res., vol. 9, no. 4, pp. 235-245, 2019.

**9.**S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions", Neurocomputing, vol. 239, pp. 39-57, 2017.

**10.**R. Li, J. Hu and S. Yang, "Deep Gated Recurrent Unit Convolution Network for Radio Signal Recognition", Int. Conf. Commun. Technol. Proceedings, pp. 159-163, 2019.

**11.**X. Zhang and T. Luo, "A RNN decoder for channel decoding under correlated noise", 2019 IEEE/CIC Int. Conf. Commun. Work. China ICCC Work. 2019, pp. 30-35, 2019.

**12.**J. Li and Y. Shen, "Image describing based on bidirectional LSTM and improved sequence sampling", 2017 IEEE 2nd Int. Conf. Big Data Anal. ICBDA 2017, pp. 735-739, 2017.

**13.**M. Razzaghi, G. J. McLachan and K. E. Basford, "Mixture Models", Technometrics, vol. 33, no. 3, pp. 365, 1991.

**14.**S. Chan and P. Treleaven, Continuous Model Selection for Large-Scale Recommender Systems, Elsevier Inc., vol. 33, 2015.

**15.**accuracy", Int. J. Forecast., vol. 22, no. 4, pp. 679-688, 2006.

**16.**R. D. Peng, Reproducible Research in Pattern Recognition, vol. 10214, December 2017.

[17] T. E. Koker and D. Koutmos, ‘‘Cryptocurrency trading using machine learning,’’ J. Risk Financial Manage., vol. 13, no. 8, p. 178, Aug. 2020.

[18] R. Gupta, A. Shukla, and S. Tanwar, ‘‘BATS: A blockchain and AI-empowered drone-assisted telesurgery system towards 6G,’’ IEEE Trans. Netw. Sci. Eng., early access, Dec. 8, 2020, doi: 10.1109/ TNSE.2020.3043262.

[19] P. Haridas, G. Chennupati, N. Santhi, P. Romero, and S. Eidenbenz, ‘‘Code characterization with graph convolutions and capsule networks,’’ IEEE Access, vol. 8, pp. 136307–136315, 2020.

[20] B. Anush. Comparative and Informative Characteristic of the Legal Regulation of the Blockchain and Cryptocurrency: State and Prospects. Accessed: 2021. [Online]. Available: https://www.annalsofrscb. ro/index.php/journal/article/view/2006

[21] M. Pravdiuk. International Experience of Cryptocurrency Regulation. Accessed: Jun. 4, 2021. [Online]. Available: https://cyberleninka.ru/article/ n/international-experience-of-cryptocur%rency-regulation

[22] S. Shanaev, S. Sharma, B. Ghimire, and A. Shuraeva, ‘‘Taming the blockchain beast? Regulatory implications for the cryptocurrency market,’’ Res. Int. Bus. Finance, vol. 51, Jan. 2020, Art. no. 101080. [23] V. Derbentsev, N. Datsenko, O. Stepanenko, and V. Bezkorovainyi, ‘‘Forecasting cryptocurrency prices time series using machine learning approach,’’ SHS Web Conf., vol. 65, Jan. 2019, Art. no. 02001. [24] Liew. The Case for Bitcoin for Institutional Investors: Bubble Investing or Fundamentally Sound. Accessed: 2017. [Online]. Available: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3082808>

[25] G. Gomes, L. Dias, and M. Correia, ‘‘CryingJackpot: Network flows and performance counters against cryptojacking,’’ in Proc. IEEE 19th Int. Symp. Netw. Comput. Appl. (NCA), Nov. 2020, pp. 1–10. [26] T. Wang. When Blockchain Meets AI: Optimal Mining Strategy Achieved by Machine Learning. Accessed: 2021. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22375>

[27] E. Sin and L. Wang, ‘‘Bitcoin price prediction using ensembles of neural networks,’’ in Proc. 13th Int. Conf. Natural Comput., Fuzzy Syst. Knowl. Discovery (ICNC-FSKD), Jul. 2017, pp. 666–671.

[28] A. Radityo, Q. Munajat, and I. Budi, ‘‘Prediction of bitcoin exchange rate to American dollar using artificial neural network methods,’’ in Proc. Int. Conf. Adv. Comput. Sci. Inf. Syst. (ICACSIS), Oct. 2017, pp. 433–438.

[29] N. A. Hitam, A. R. Ismail, and F. Saeed, ‘‘An optimized support vector machine (SVM) based on particle swarm optimization (PSO) for cryptocurrency forecasting,’’ Proc. Comput. Sci., vol. 163, pp. 427–433, Jan. 2019.

[30] P. M., A. Sharma, V. V., V. Bhardwaj, A. P. Sharma, R. Iqbal, and R. Kumar, ‘‘Prediction of the price of Ethereum blockchain cryptocurrency in an industrial finance system,’’ Comput. Electr. Eng., vol. 81, Jan. 2020, Art. no. 106527.

[31] S. Tandon, S. Tripathi, P. Saraswat, and C. Dabas, ‘‘Bitcoin price forecasting using LSTM and 10-fold cross validation,’’ in Proc. Int. Conf. Signal Process. Commun. (ICSC), Mar. 2019, pp. 323–328