Group 47

Investopedia

Jehan Desai (21BAI10034)
Akshat Pathak (21BAI10053)
Garv Malik (21BAI10070)
Ishaan Arora (21BAI10081)
Anish Sheikh (21BAI10299)

Agenda

- Problem Statement
- Scope of the Project
- System Requirememts
- Process Flow
- Application
- Contibution of each member

Problem Statement

- The stock market appears in the news every day. You hear about it every time it reaches a new high or a new low. The rate of investment and business opportunities in the Stock market can increase if an efficient algorithm could be devised to predict the short term price of an individual stock.

Scope of the Project

At the present time there are lots of applications available which can predict gold, house, shares, etc. So, instead of building different applications to predict these stock we are making an application which can predict all these within a single application. This saves the user more memory in their devices and also the user return to the application is maintained.

Requirements

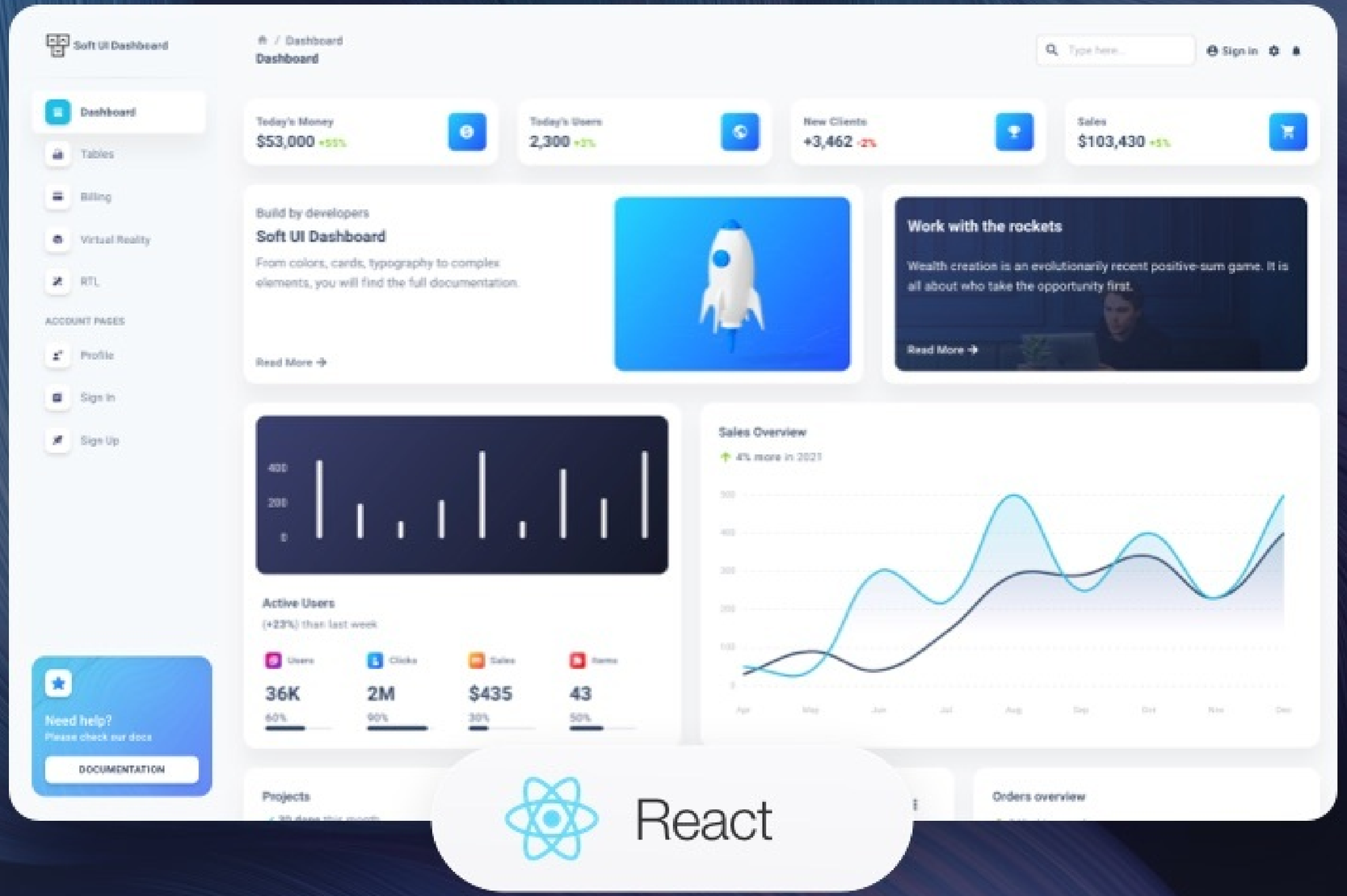
Software

- IDE:
  - Google Colab (Backend)
  - VS code (Frontend)
- Python:
  - TensorFlow/Keras
  - Scikit Learn
- ReactJs:
  - ES6 Modules
  - Bootstrap
  - Tensorflow.js + React.js
- CLI

Hardware

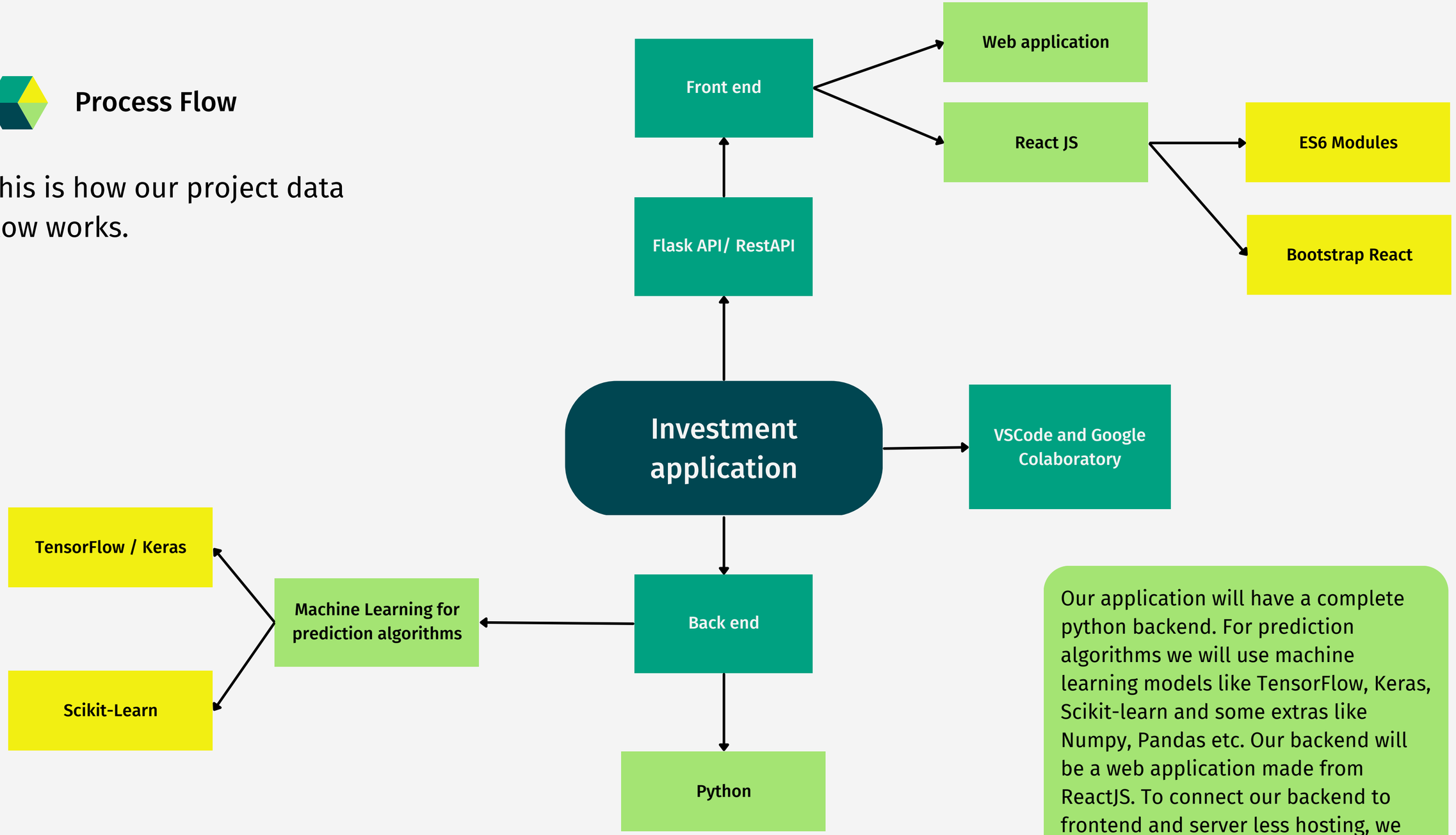
- Laptop
- RAM: 8GB
- Graphic card:
  - Given by Google Colab (K80 GPU and 12 GB of RAM)

UX Design


Soft UI Dashboard
Dashboard
Dashboard
Tables
Billing
Virtual Reality
RTL
ACCOUNT PAGES
Profile
Sign In
Sign Up
Need help?
Please check our docs
DOCUMENTATION
Dashboard
Type here...
Sign In
Today's Money
$53,000 +55%
Today's Users
2,300 +3%
New Clients
+3,462 -2%
Sales
$103,430 +5%
Build by developers
Soft UI Dashboard
From colors, cards, typography to complex elements, you will find the full documentation.
Read More
Work with the rockets
Wealth creation is an evolutionarily recent positive-sum game. It is all about who take the opportunity first.
Read More
Active Users
(+23%) than last week
Users
Clicks
Sales
Items
36K
2M
$435
43
60%
90%
30%
50%
Sales Overview
4% more in 2021
Projects
Orders overview
React

Process Flow

This is how our project data flow works.



Our application will have a complete python backend. For prediction algorithms we will use machine learning models like TensorFlow, Keras, Scikit-learn and some extras like Numpy, Pandas etc. Our backend will be a web application made from ReactJS. To connect our backend to frontend and server less hosting, we are using API's.

Usability

- This application has a major usability criteria. We have loads and loads of new businesses everyday and there are new people in the market everyday looking for investment opportunities. Our application gives them a platform where majority users can find the opportunities to get the highest return from their investments. Using our prediction algorithm, customers can get all of the most common and low stake asset market predictions.
- This product will cover majority of the population as everyone wants to invest in the market nowadays.
- Many companies also have an issue of returning users. This means that how many people return to the application daily. Many apps start strong but soar down as they go forward. With more than 1 types of assets, we will have different users who like to invest in different markets. This maintains our returning users issue and keeps the application alive.

Review 2

Stock Prediction Model

Step 1: Import the modules

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Step 2: Defining Variables

```
# Load the stock data
df = pd.read_csv('NFTY.csv')

# Extract the closing price as the target variable
y = df['close']

# Extract the features
X = df[['open', 'high', 'low', 'volume']]

```

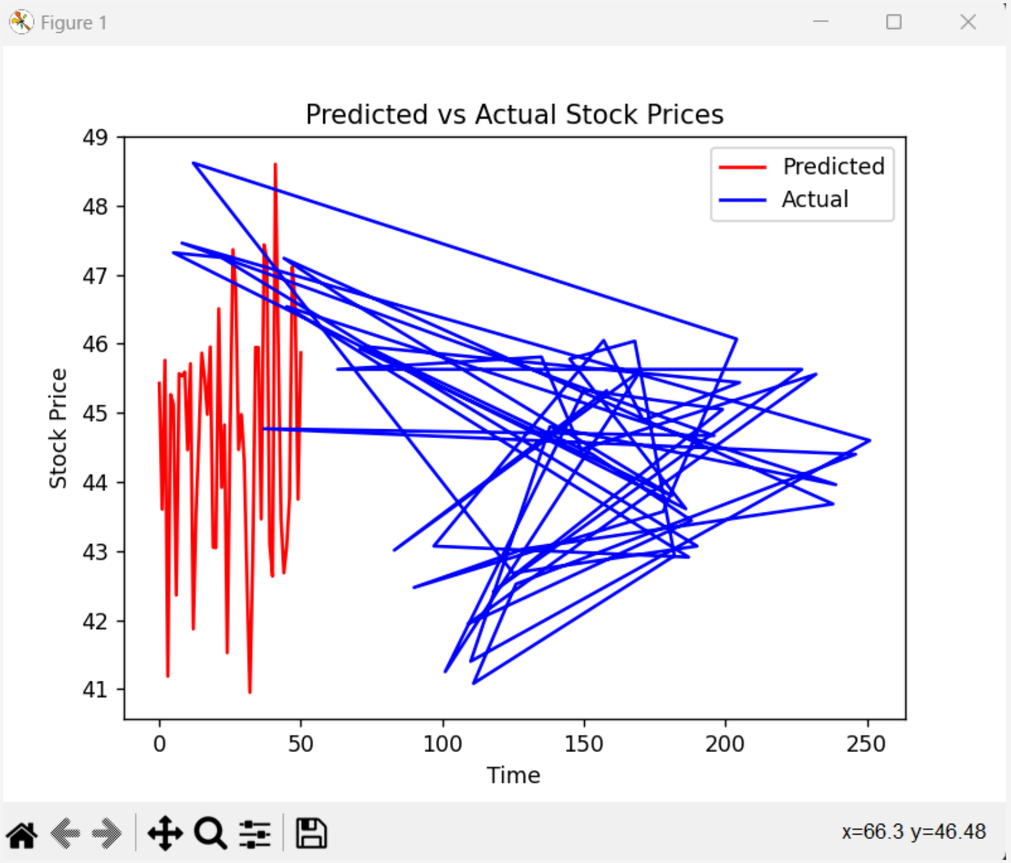
Step 3: Training the model

```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Train the linear regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predict the stock prices
y_pred = regressor.predict(X_test)
```

Step 4: Plotting the graph

```
# Plot the predicted stock prices
plt.plot(y_pred, color='red', label='Predicted')
# Plot the actual stock prices
plt.plot(y_test, color='blue', label='Actual')
# Add labels and a title
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.title('Predicted vs Actual Stock Prices')
plt.legend()
# Show the plot
plt.show()
```

Output



Note: that the plot of the actual stock prices in this code may appear messy. This is because the plt.plot() function is being used to plot the y_test data, which is a series of individual stock prices over time. Since there are a large number of data points, the plot may appear cluttered and difficult to interpret.

Stock Prediction Model (Updated)

NFTY.py

Models > NFTY.py > ...

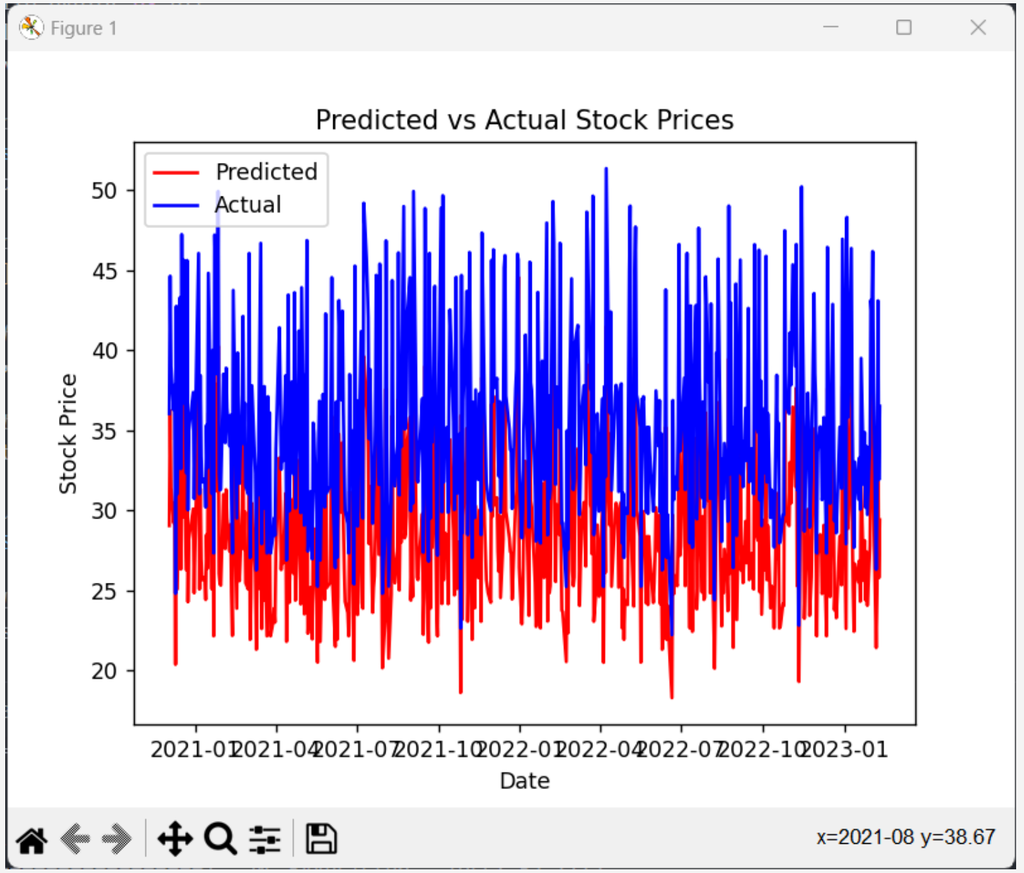
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras

# Load the stock data
df = pd.read_csv('NFTY.csv')
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')

# Extract the closing price as the target variable
y = df['Close'].values

# Extract the features
X = df[['Open', 'High', 'Low', 'Volume']].values

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create a sequential model
model = keras.Sequential()

# Add a dense layer
model.add(keras.layers.Dense(1, input_shape=(X_train.shape[1],)))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32)

# Predict the stock prices
y_pred = model.predict(X_test)

model.save('NFTY.h5')
```

NFTY.py

Models > NFTY.py > ...

```python
# Plot the predicted stock prices
plt.plot(df['Date'][len(X_train):], y_pred, color='red', label='Predicted')
# Plot the actual stock prices
plt.plot(df['Date'][len(X_train):], y_test, color='blue', label='Actual')
# Add labels and a title
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.title('Predicted vs Actual Stock Prices')
plt.legend()
# Show the plot
plt.show()

def predict_price(date):
    # convert the date string to a datetime object
    date = pd.to_datetime(date, format='%d-%m-%Y')
    # find the index of the date in the dataframe
    index = df[df['Date'] == date].index[0]
    # use the index to retrieve the features for the date
    features = X[index].reshape(1, -1)
    # use the model to predict the stock price for the date
    price = model.predict(features)[0][0]
    return price

date = '10-02-2023'
price = predict_price(date)
print(f'The predicted stock price for {date} is {price:.2f}')
```

Output



Note: that the plot of the actual stock prices in this code may appear messy. This is because the plt.plot() function is being used to plot the y_test data, which is a series of individual stock prices over time. Since there are a large number of data points, the plot may appear cluttered and difficult to interpret.

Gold prediction model

```
import pandas as pd
import numpy as np
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

Modules used

```
df = pd.read_csv("gold_price.csv", parse_dates=True, index_col='Date')
```

```
df.head()
```

```
df['Return'] = df['USD (PM)'].pct_change() * 100
df['Lagged_Return'] = df.Return.shift()
df = df.dropna()
train = df['2001':'2018']
test = df['2019']
# Create train and test sets for dependent and independent variables
X_train = train["Lagged_Return"].to_frame()
y_train = train["Return"]
X_test = test["Lagged_Return"].to_frame()
y_test = test["Return"]
```

```
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

Training the model

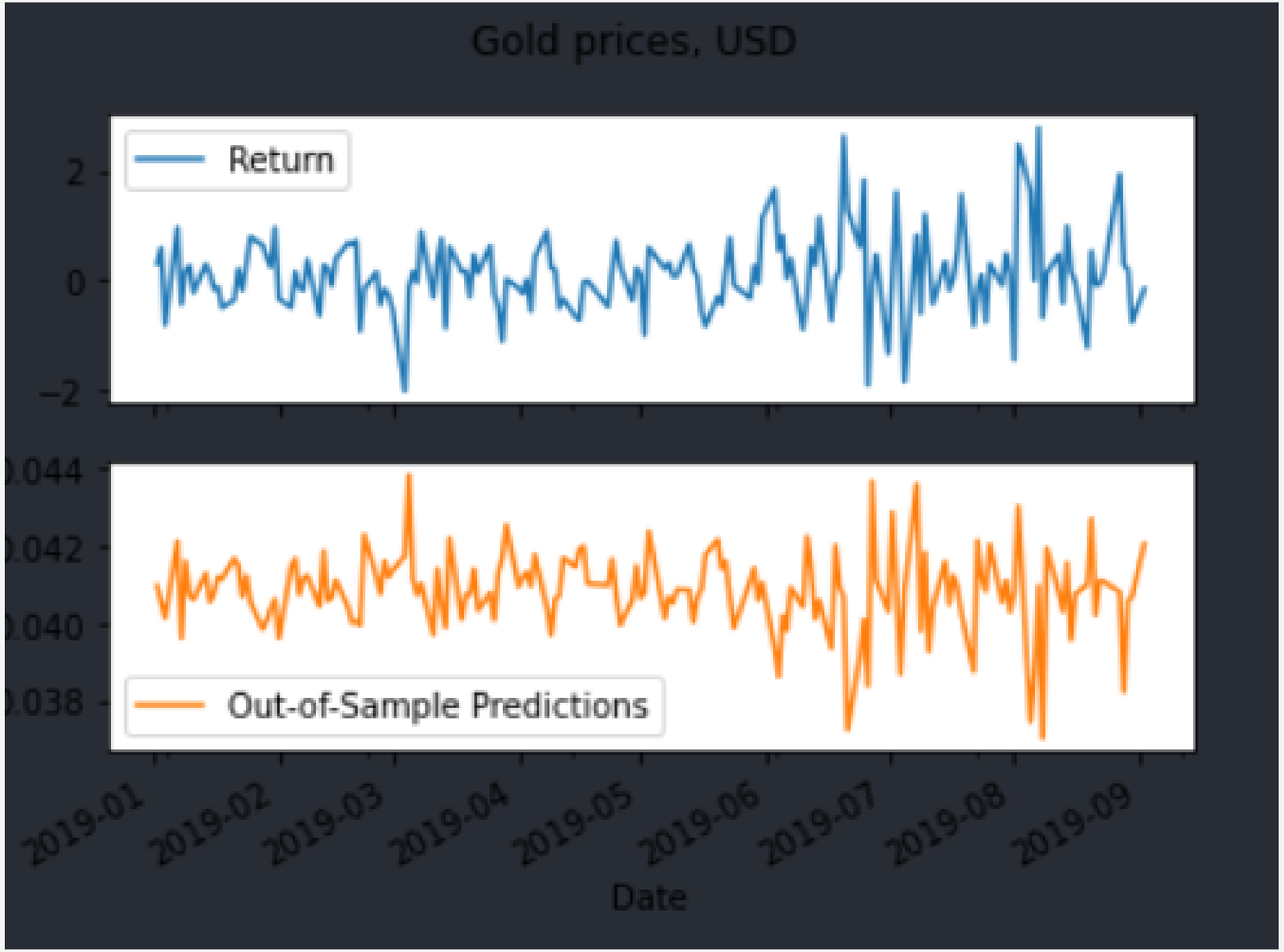
```
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

Saving the model checkpoints and printing the graph

```
import matplotlib.pyplot as plt
out_of_sample_results = y_test.to_frame()
```

```
out_of_sample_results["Out-of-Sample Predictions"] = model.predict(X_test)
out_of_sample_results.plot(subplots=True, title='Gold prices, USD')
plt.show()
```

Output


Gold prices, USD
Return
2
0
−2
.044
.042
.040
.038
Out-of-Sample Predictions
2019-01
2019-02
2019-03
2019-04
2019-05
2019-06
2019-07
2019-08
2019-09
Date

Review 3

Home webpage

```html
<!DOCTYPE html>
<html>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="stylesheet.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Fredoka+One&family=Play&display=swap" rel="stylesheet">

    <body>
        <nav class="nav">
            <div class="container">
                <div class="logo">
                    <a href="#"><img src="logo.png", height=60px/></a>
                </div>
                <div id="mainListDiv" class="main_list">
                    <ul class="navlinks">
                        <li><a href="home.html">Home</a></li>
                        <li><a href="google.html">Google</a></li>
                        <li><a href="NFTY.html">NFTY</a></li>
                        <li><a href="Microsoft.html">Microsoft</a></li>
                        <li><a href="Apple.html">Apple</a></li>
                        <li><a href="Gold.html">Gold</a></li>
                        <li><a href="Housing.html">Housing</a></li>
                    </ul>
                </div>
                <span class="navTrigger">
                    <i></i>
                    <i></i>
                    <i></i>
                </span>
            </div>
        </nav>

        <section class="home">
        </section>
        <div style="height: 1000px">
            <!-- just to make scrolling effect possible -->
            <h2 class="myH2">Welcome to INVESTOPEDIA</h2>
            <p class="myP">This is an all in one website for investers.</p>
            <p class="myP">We have a collection of the predictions of the most common stocks people invest in and other assets such as
            <br>
            <br>
            <br>
            <br>
            <p class="myP">Wondering how to use our platform? Here's how you can uncderstand our product:</p>
            <p></p>
            <br>
            <br>
            <ol class="myP">
                <li>We have a range of assets available in our website. Nowadays the most common stocks that people are investing in ar
                <li>Choose anyone of the following stocks and go to their respective webpage. There you will see a graph of past prices
                <li>Our model predicts the prices of tomorrow and gives you a recommendation of wether to invest or not. If it is red,
                <li>Once you see the eye opening predictions, you can read through the small jist or reasoning as to why or why not inv
                <li>SEE!! ITS THAT EASY!!</li>
            </ol>
            <br>
            <br>
            <br>
            <br>
            <p class="myP">WE HOPE YOU FIND THIS USEFULL</p>
            <p class="myP"></p>
                <p class="myP">
            </p>
        </div>

    <!-- Jquery needed -->
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
        <script src="js/scripts.js"></script>

    <!-- Function used to shrink nav bar removing paddings and adding black background -->
        <script>
            $(window).scroll(function() {
                if ($(document).scrollTop() > 50) {
                    $('.nav').addClass('affix');
                    console.log("OK");
                } else {
                    $('.nav').removeClass('affix');
                }
            });
        </script>

    <footer>
        <div class="footer">
        <div class="row">
        <a href="#"><i class="fa fa-facebook"></i></a>
        <a href="#"><i class="fa fa-instagram"></i></a>
        <a href="#"><i class="fa fa-youtube"></i></a>
        <a href="#"><i class="fa fa-twitter"></i></a>
        </div>

        <div class="row">
        <ul>
        <li><a href="#">Contact us</a></li>
        <li><a href="#">Our Services</a></li>
        <li><a href="#">Privacy Policy</a></li>
        <li><a href="#">Terms & Conditions</a></li>
        <li><a href="#">Our product</a></li>
        </ul>
        </div>

        <div class="row">
        </div>
        </div>
    </footer>

    </body>
</html>
```

```css
footer {
  text-align: center;
  padding: 3px;
  background-color: black;
  color: white;
}

body{
  margin:0;
  overflow-x:hidden;
  }

  .footer{
  background:#000;
  padding:1px 0px;
  font-family: 'Play', sans-serif;
  text-align:center;
  }

  .footer .row{
  width:100%;
  margin:1% 0%;
  padding:0.6% 0%;
  color:gray;
  font-size:0.8em;
  }

  .footer .row a{
  text-decoration:none;
  color:gray;
  transition:0.5s;
  }

  .footer .row a:hover{
  color:#fff;
  }

  .footer .row ul{
  width:100%;
  }

  .footer .row ul li{
  display:inline-block;
  margin:0px 30px;
  }

  .footer .row a i{
  font-size:2em;
  margin:0% 1%;
  }

  @media (max-width:720px){
  .footer{
  text-align:left;
  padding:5%;
  }
  .footer .row ul li{
  display:block;
  margin:10px 0px;
  text-align:left;
  }
  .footer .row a i{
  margin:0% 3%;
  }
  }
```

```css
.myH2 {
    text-align:center;
    font-size: 4rem;
    color: white;
    font-size: 80px;
}
.myP {
    text-align: center;
    padding-left:15%;
    padding-right:15%;
    font-size: 25px;
    color: white;
}
.myP2 {
    text-align: center;
    padding-left:15%;
    padding-right:15%;
    font-size: 20px;
    font-weight: bold;
    color: white;

}
@media all and (max-width:700px){
    .myP {
        padding:2%;
    }
}

form {
    padding: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
    text-align: center;
  }

  label {
    margin-top: 10px;
  }

  input[type="file"] {
    margin-top: 10px;
    padding: 10px;
    border: 1px solid lightgray;
    border-radius: 5px;
    display: inline-block;
  }

  button[type="submit"] {
    margin-top: 20px;
    padding: 10px 20px;
    background-color: green;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
  }

  .imgcenter {
    display: block;
    margin-left: auto;
    margin-right: auto;
    height: 500px;
    width: 950px;
    border-radius: 3%;
  }
```

```css
@keyframes outT {
    0% {
        transform: translateY(0px) rotate(0deg);
    }
    50% {
        transform: translateY(9px) rotate(0deg);
    }
    100% {
        transform: translateY(9px) rotate(135deg);
    }
}

@-webkit-keyframes inBtm {
    0% {
        -webkit-transform: translateY(0px) rotate(0deg);
    }
    50% {
        -webkit-transform: translateY(-9px) rotate(0deg);
    }
    100% {
        -webkit-transform: translateY(-9px) rotate(135deg);
    }
}

@keyframes inBtm {
    0% {
        transform: translateY(0px) rotate(0deg);
    }
    50% {
        transform: translateY(-9px) rotate(0deg);
    }
    100% {
        transform: translateY(-9px) rotate(135deg);
    }
}

@-webkit-keyframes outBtm {
    0% {
        -webkit-transform: translateY(0px) rotate(0deg);
    }
    50% {
        -webkit-transform: translateY(-9px) rotate(0deg);
    }
    100% {
        -webkit-transform: translateY(-9px) rotate(135deg);
    }
}

@keyframes outBtm {
    0% {
        transform: translateY(0px) rotate(0deg);
    }
    50% {
        transform: translateY(-9px) rotate(0deg);
    }
    100% {
        transform: translateY(-9px) rotate(135deg);
    }
}

.affix {
    padding: 0;
    background-color: #111;
}
```

```css
@keyframes inM {
    50% {
        transform: rotate(0deg);
    }
    100% {
        transform: rotate(45deg);
    }
}

@-webkit-keyframes outM {
    50% {
        -webkit-transform: rotate(0deg);
    }
    100% {
        -webkit-transform: rotate(45deg);
    }
}

@keyframes outM {
    50% {
        transform: rotate(0deg);
    }
    100% {
        transform: rotate(45deg);
    }
}

@-webkit-keyframes inT {
    0% {
        -webkit-transform: translateY(0px) rotate(0deg);
    }
    50% {
        -webkit-transform: translateY(9px) rotate(0deg);
    }
    100% {
        -webkit-transform: translateY(9px) rotate(135deg);
    }
}

@keyframes inT {
    0% {
        transform: translateY(0px) rotate(0deg);
    }
    50% {
        transform: translateY(9px) rotate(0deg);
    }
    100% {
        transform: translateY(9px) rotate(135deg);
    }
}

@-webkit-keyframes outT {
    0% {
        -webkit-transform: translateY(0px) rotate(0deg);
    }
    50% {
        -webkit-transform: translateY(9px) rotate(0deg);
    }
    100% {
        -webkit-transform: translateY(9px) rotate(135deg);
    }
}
```

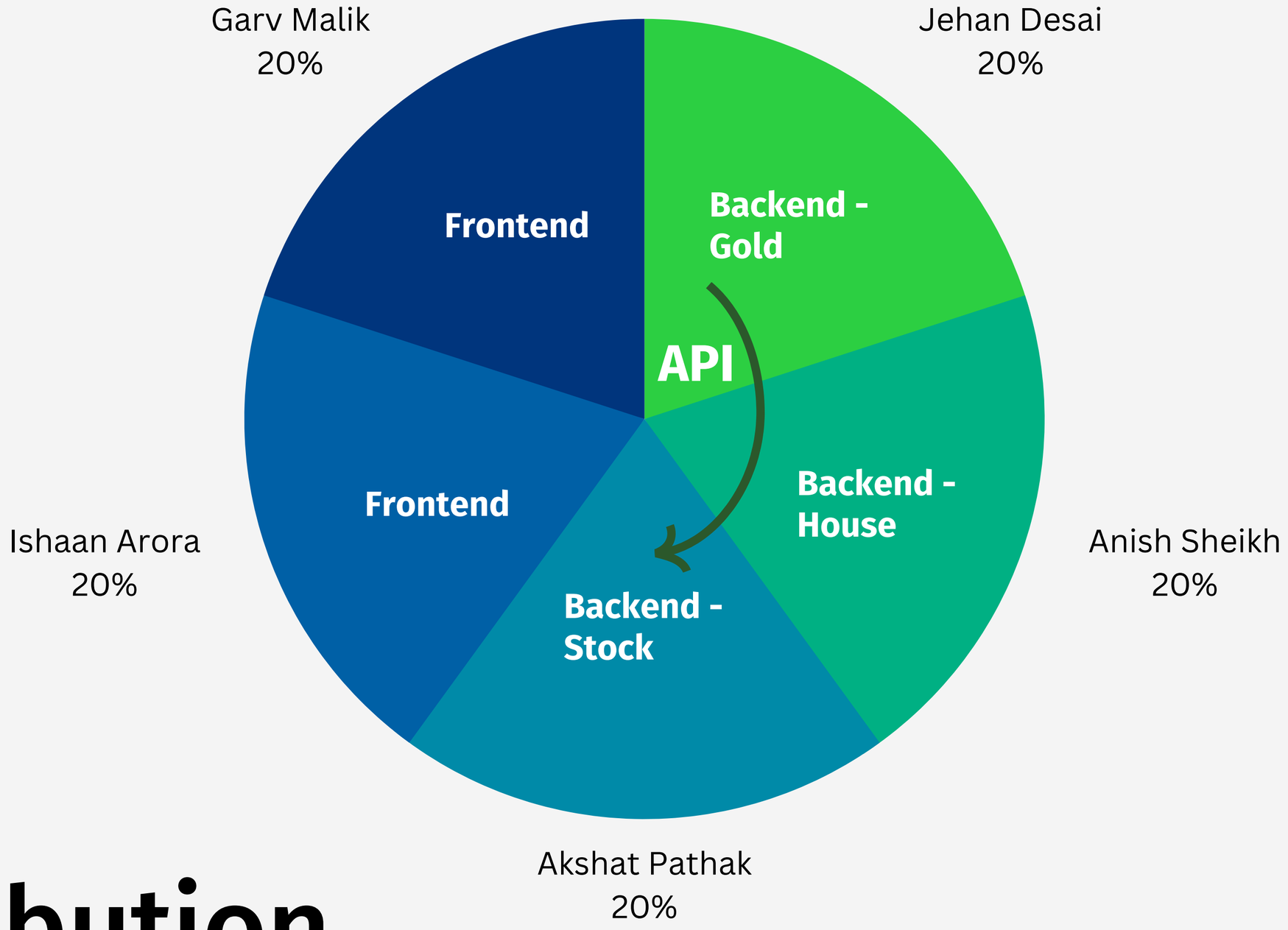
```css
@import url('https://fonts.googleapis.com/css?family=Quicksand:400,500,700');
html,
body {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: "Quicksand", sans-serif;
    font-size: 62.5%;
    font-size: 10px;
    background: rgb(2,0,36);
    background: radial-gradient(circle, rgba(2,0,36,1) 0%, rgba(0,39,194,1)
}
/*-- Inspiration taken from abdo steif -->
/* --> https://codepen.io/abdosteif/pen/bRoyMb?editors=1100*/

/* Navbar section */

.nav {
    width: 100%;
    height: 65px;
    position: fixed;
    line-height: 65px;
    text-align: center;
}

.nav div.logo {
    float: left;
    width: auto;
    height: auto;
    padding-left: 3rem;
}

.nav div.logo a {
    text-decoration: none;
    color: #fff;
    font-size: 2.5rem;
}

.nav div.logo a:hover {
    color: #c5fad5;
}

.nav div.main_list {
    height: 65px;
    float: right;
}

.nav div.main_list ul {
    width: 100%;
    height: 65px;
    display: flex;
    list-style: none;
    margin: 0;
    padding: 0;
}

.nav div.main_list ul li {
    width: auto;
    height: 65px;
    padding: 0;
    padding-right: 3rem;
}

.nav div.main_list ul li a {
    text-decoration: none;
    color: #fff;
    line-height: 65px;
    font-size: 2.4rem;
}
```

CSS file for our webpage

```
from sklearn.datasets import load_boston
import numpy as no
import pandas as pd
import matplotlib.pyplot as plt
df = load_boston()
dataset = pd.DataFrame(df.data)
dataset.head()
dataset.shape
target = df.target
target.shape
dataset.isnull().sum()
dataset.columns = df.feature_names
dataset["Price"] = target
X = dataset.loc[:, : 'LSTAT']   # independent features
y = dataset.loc[:, 'Price']    # dependent features
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score    # Cross Validation
lin_regression = LinearRegression()
mse = cross_val_score(lin_regression, X, y, scoring='neg_mean_squared_error', cv=5)
print(type(mse))
mse = np.mean(mse)    # we will get the five values, then we calculate the mean
print(mse)
#ridge regression
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV    # to find the value of 'lambda'

ridge = Ridge()
parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-3, 1e-2, 1, 5, 10, 20, 30, 35, 40, 45, 50, 55, 100]}
ridge_regressor = GridSearchCV(ridge, parameters, scoring='neg_mean_squared_error', cv=5)
ridge_regressor.fit(X, y)
#Lasso regression
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV

lasso = Lasso()
parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-3, 1e-2, 1, 5, 10, 20, 30, 35, 40, 45, 50, 55, 100]}
lasso_regressor = GridSearchCV(lasso, parameters, scoring='neg_mean_squared_error', cv=5)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
prediction_lasso = lasso_regressor.predict(X_test)
prediction_ridge = ridge_regressor.predict(X_test)
```

House prediction model using 3 different regressions

1. Linear regression
2. Ridge regression
3. Lasso regression



Contribution

THANK YOU