TEAM SMURFS

BOND PRICE PREDICTION
&
PORTFOLIO RECOMMENDATION OF TOP 5 BONDS WITH HIGHEST RETURN/VOLATILITY RATIO

Domain : Finance

INTRODUCTION

BOND PRICE

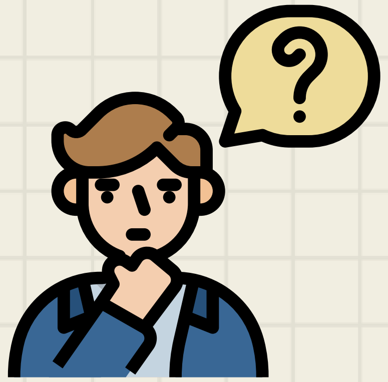
Reflection of extremely complex market interactions and policies making it difficult to predict

dearth of relevant information

accuracy

time

PROBLEM DEFINITION



Bond prices are dynamic in the market and can be influenced by different input features:

- interest rate(linked inversely)
- rating of the issuer
- bond's maturity
- duration and so on.

All of the input features should be included in the prediction of bond price with the use of machine learning to obtain high accuracy.

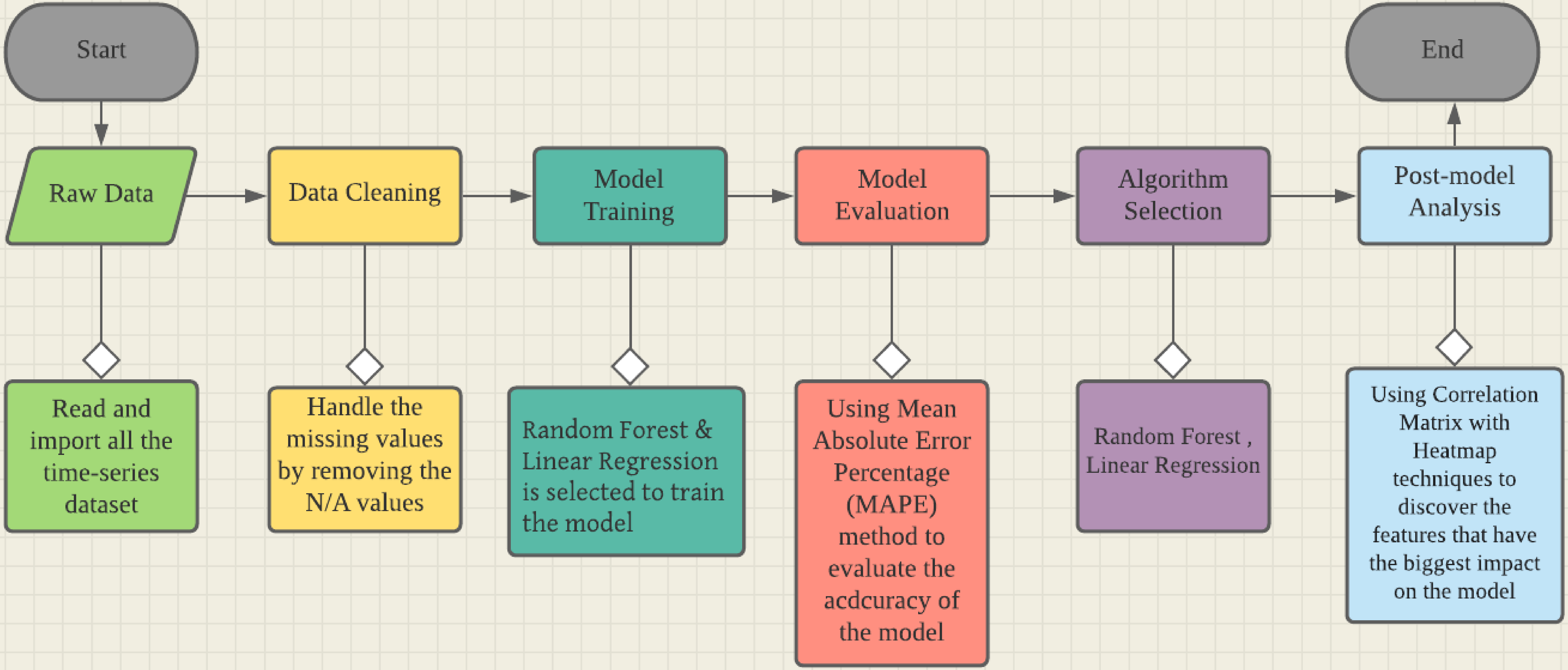
OBJECTIVE & GOALS

We use a dataset describing a large number of bonds along with other relevant descriptive metrics to :

1. Predict the bond's price in next month
2. Calculate the bond return and volatility
3. Create a portfolio of the Top 5 Bond that gives the highest Return/Volatility ratio.

IMPLEMENTATION

pre-Stage: Build machine learning model



IMPLEMENTATION

pre-Stage: Build machine learning model (Data preparation phase)

Step 1: Filter and only include rating between AAA to AA3

```
df2 = df[df['RATING'].str.contains("AA")].reset_index(drop=True)
```

IMPLEMENTATION

Step 2: Feature Engineering

[6]

```
for col in ['VALUE DATE','PREVIOUS PAYMENT DATE','ISSUE DATE','MATURITY DATE','NEXT PAYMENT DATE']:
    df2[col] = pd.to_datetime(df2[col])
```

[7]

```
df2['PREVIOUS PAYMENT DATE for calculation'] = df2['PREVIOUS PAYMENT DATE'].copy()
df2.loc[df2['PREVIOUS PAYMENT DATE for calculation'].isnull(),'PREVIOUS PAYMENT DATE for calculation']=df2.loc[df2['PREVIOUS PAYMENT DATE for calculation'].isnull()]['ISSUE DATE'
```

[8]

```
df2["Value-Prev"] = df2['VALUE DATE'] - df2['PREVIOUS PAYMENT DATE for calculation']
df2["Value-Prev"] = df2["Value-Prev"].astype(str).str.replace(" days","").astype(int)
```

[9]

```
df2.dropna(subset=['PREVIOUS PAYMENT DATE','NEXT PAYMENT DATE'],inplace=True)
df2["Payment Days diff"] = df2['NEXT PAYMENT DATE'] - df2['PREVIOUS PAYMENT DATE']
df2["Payment Days diff"] = df2["Payment Days diff"].astype(str).str.replace(" days","").astype(int)
```

IMPLEMENTATION

Step 2: Feature Engineering

```
df2['Year'] = df2['Year_Month'].str[:4].astype(int)
df2['Month'] = df2['Year_Month'].str[4:].astype(int)
df2['Day'] = 1

df2['Date'] = pd.to_datetime(df2[["Year","Month","Day"]])
```

[10]

```
thisdict = {
  "ACTACT": 1,
  "ACT365": 2,
  "ACTBOTH": 3
}

df2['DAY COUNT BASIS (INT)']= df2['DAY COUNT BASIS'].map(thisdict)
df2 = df2[df2['DAY COUNT BASIS (INT)']!=3]
```

[11]

IMPLEMENTATION

Step 3: Calculations

```
nominal = 5000000

df2["ACCRUED INTEREST 1"] = nominal*(df2['NEXT COUPON RATE']/100)*(df2["Value-Prev"]/df2["Payment Days diff"])
df2["ACCRUED INTEREST 2"] = nominal*(df2['NEXT COUPON RATE']/100)*(df2["Value-Prev"]/365)

df2['ACCRUED INTEREST'] = df2["ACCRUED INTEREST 1"].copy()
df2.loc[df2['DAY COUNT BASIS (INT)']==2,'ACCRUED INTEREST'] = df2.loc[df2['DAY COUNT BASIS (INT)']==2]["ACCRUED INTEREST 2"]
```

[12]

```
df2["TERM OF MATURITY"]= (df2['MATURITY DATE'] - df2['VALUE DATE'])
df2["TERM OF MATURITY"] = df2["TERM OF MATURITY"].astype(str).str.replace(" days","").astype(int)
df2["TERM OF MATURITY"] = df2["TERM OF MATURITY"]/365
```

[13]

Step 4: Define features and lable to fit into the Machine Learning model

```
X_col = ['EVAL MID YIELD','EVAL MID PRICE','ACCRUED INTEREST',"TERM OF MATURITY",'MODIFIED DURATION']
y_col = 'EVAL MID PRICE M1'
X = no_sep_oct[X_col]
y = no_sep_oct[y_col]
```

[17]

IMPLEMENTATION

Step 5: Split data into train, test & validation sets before we build the model

Train set, Test set, Validation set

```
# Using Skicit-learn to split data into training and testing sets
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size = 0.25, random_state = 42)
```

[21]

```
print('Training X Shape:', train_X.shape)
print('Training y Shape:', test_X.shape)
print('Testing X Shape:', train_y.shape)
print('Testing y Shape:', test_y.shape)
```

[22]

```
Training X Shape: (14461, 5)
Training y Shape: (4821, 5)
Testing X Shape: (14461,)
Testing y Shape: (4821,)
```

IMPLEMENTATION

Random Forest Algorithm
(Non-parametric model)

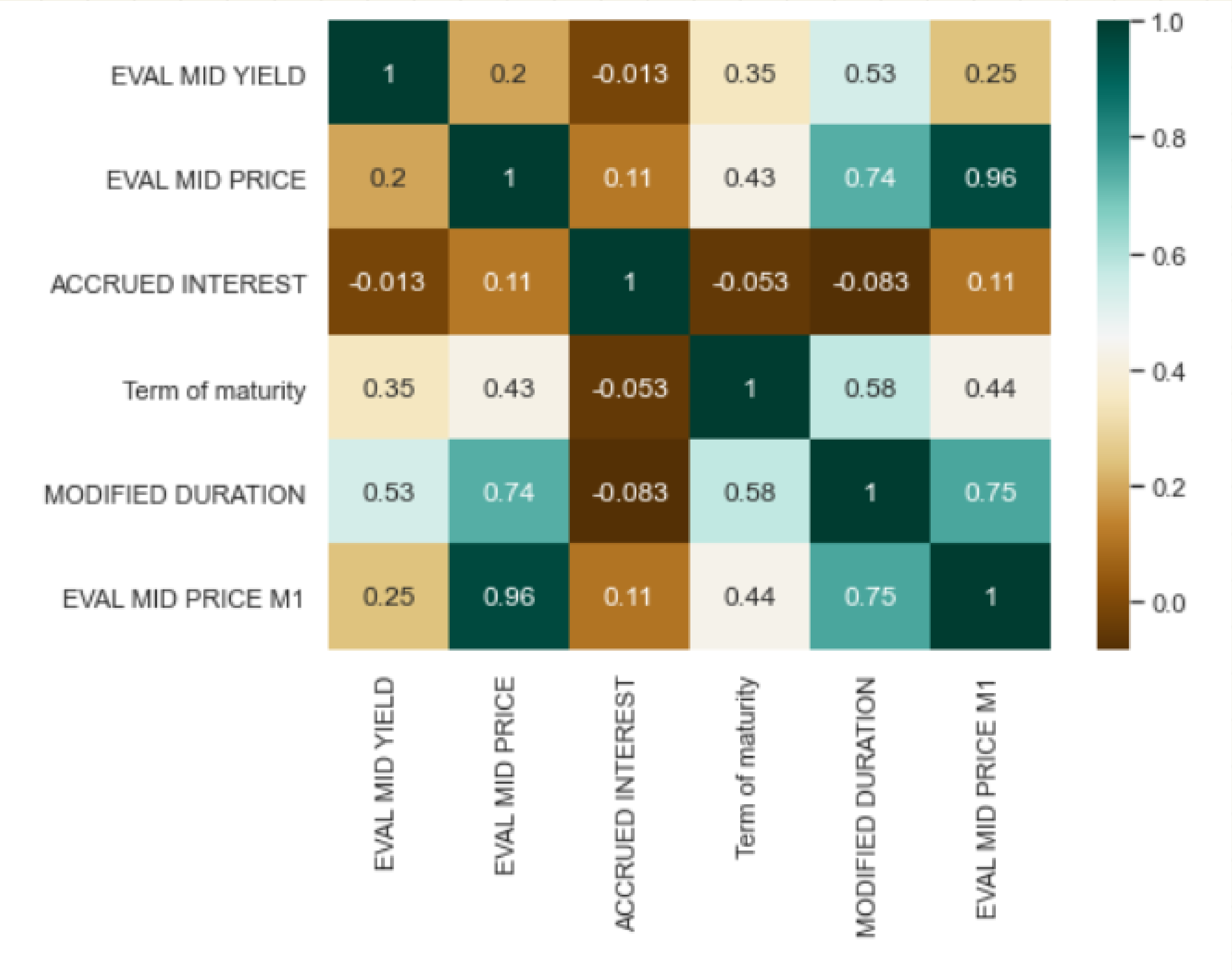
```
Random Forest:

Mean Absolute Error Test: 0.8
Mean Absolute Error Sep, Oct: 0.45

Accuracy test: 99.27 %.
Accuracy Sep,Oct: 99.6 %.

coefficient of determination: 0.94
coefficient of determination Sep,Oct: 0.98
```

Linear Regression Algorithm
(Parametric model)

```
Linear Regression:

Mean Absolute Error: 0.86
Mean Absolute Error Sep, Oct: 0.39

Accuracy: 99.21 %.
Accuracy Sep,Oct: 99.65 %.

coefficient of determination: 0.93
coefficient of determination Sep,Oct: 0.99
```

IMPLEMENTATION

Step 6: Post-model analysis



IMPLEMENTATION

Stage 1: Bond Price prediction

Predict the bond price by using the machine learning model we have trained

```
forecast['M+1 BOND PRICE'] = model.predict(forecast[X_col])
```

IMPLEMENTATION

Stage 2: Bond Return and Volatility Calculation

We used the given formula to calculate the bond return and the volatility.

Step 1: To calculate Bond Return based on M+1 Bond Price Prediction

```
forecast['M+1 YIELD INCOME'] = forecast['M+1 BOND PRICE']/forecast['COUPON FREQUENCY']
forecast['M+1 ROLL DOWN RETURN'] = (forecast['M+1 BOND PRICE'] - forecast['EVAL MID PRICE BEG'])/forecast['EVAL MID PRICE BEG']
forecast['M0 EXPECTED CHG IN YIELD'] = (-forecast['MD']*forecast['EVAL MID YIELD CHANGE'])+(0.5*forecast['CONVEXITY']*forecast['EVAL MID YIELD CHANGE']*forecast['EVAL MID YIELD C

forecast['M+1 BOND RETURN'] = forecast[['M+1 YIELD INCOME','M+1 ROLL DOWN RETURN','M0 EXPECTED CHG IN YIELD']].sum(1)
```

IMPLEMENTATION

Step 2: To Calculate Bond Volatility and Return/Volatility ratio based on M+1 Bond Price Prediction

```python
final_vol_df = pd.DataFrame()

for dd_ in np.sort(forecast['Date'].unique()):
    dd_m1 = datetime.strptime(str(dd_)[0:10], '%Y-%m-%d') + relativedelta(months=+1)
    print(str(dd_)[0:10])
    print(str(dd_m1)[0:10])
    print("")

    df_forecast = forecast[forecast['Date']==dd_][['STOCK CODE','M+1 BOND PRICE']]
    df_forecast.rename(columns={"M+1 BOND PRICE":"EVAL MID PRICE"},inplace=True)
    df_forecast['Date'] = pd.to_datetime(str(dd_m1)[0:10])
    print(df_forecast['Date'].unique())

    vol_df = df2[df2['Date']<=dd_][['STOCK CODE','Date','EVAL MID PRICE']].reset_index(drop=True)
    vol_df = vol_df.append(df_forecast[['STOCK CODE','Date','EVAL MID PRICE']].reset_index(drop=True),ignore_index=True)
    print(np.sort(vol_df['Date'].unique()))
    print("")

    vol_df = vol_df.groupby(['STOCK CODE']).agg({"EVAL MID PRICE":np.nanstd}).reset_index()
    vol_df['Date'] = pd.to_datetime(str(dd_)[0:10])
    vol_df.rename(columns={"EVAL MID PRICE":"EVAL MID PRICE STD DEV"},inplace=True)

    final_vol_df = final_vol_df.append(vol_df,ignore_index=True)
```

[87]

```python
forecast['M+1 BOND VOLATILITY'] = forecast['EVAL MID PRICE STD DEV'] * np.sqrt(252)
forecast['M+1 RETURN/VOLATILITY'] = forecast['M+1 BOND RETURN']/forecast['M+1 BOND VOLATILITY']

forecast['PREDICTION DATE'] = forecast['Date'] + pd.DateOffset(months=1)
```

[89]

IMPLEMENTATION

Step 3: Result

```
forecast[['STOCK CODE','Date','PREDICTION DATE','EVAL MID PRICE','M+1 BOND PRICE','EVAL MID PRICE M1','M+1 BOND RETURN','M+1 BOND VOLATILITY','M+1 RETURN/VOLATILITY']]
```

	STOCK CODE	Date	PREDICTION DATE	EVAL MID PRICE	M+1 BOND PRICE	EVAL MID PRICE M1	M+1 BOND RETURN	M+1 BOND VOLATILITY	M+1 RETURN/VOLATILITY
0	DZ012851	2019-07-01	2019-08-01	108.356	108.027006	108.273	54.010467	3.692953	14.625278
1	DZ012851	2019-08-01	2019-09-01	108.273	107.935034	107.955	54.251811	3.539705	15.326648
2	DZ012851	2019-09-01	2019-10-01	107.955	107.666397	107.650	53.826834	5.010803	10.742157
3	DZ012851	2019-10-01	2019-11-01	107.650	107.405470	107.307	53.693963	6.415245	8.369745
4	DZ012851	2019-11-01	2019-12-01	107.307	107.116320	107.016	53.546719	8.052193	6.649955
...	...	...	...	...	...	...	...	...	...
22329	UI200090	2020-10-01	2020-11-01	103.123	103.129337	NaN	51.564730	0.071130	724.934082
22330	UF200094	2020-10-01	2020-11-01	101.747	101.237113	NaN	50.613545	5.723466	8.843164
22331	UI200083	2020-10-01	2020-11-01	102.488	102.508600	NaN	51.254501	0.231230	221.660191
22332	VE200066	2020-10-01	2020-11-01	100.203	100.196005	NaN	50.103933	0.078521	638.098457
22333	VE200062	2020-10-01	2020-11-01	100.261	99.803914	NaN	49.957418	5.130783	9.736803

22334 rows × 9 columns

IMPLEMENTATION

Stage 3: Portfolio Recommendation

We display the top 5 bonds with the highest return/volatility in next month

```
forecast[forecast['PREDICTION DATE']=='2020-11-01'].sort_values(["M+1 RETURN/VOLATILITY"],ascending=False)[['STOCK CODE','ISIN CODE','STOCK NAME','PREDICTION DATE',
                                                                    'M+1 BOND PRICE', 'M+1 YIELD INCOME',
                                                                   'M+1 ROLL DOWN RETURN','M0 EXPECTED CHG IN YIELD',
                                                                   'M+1 BOND RETURN',
                                                                    'M+1 BOND VOLATILITY',
                                                                    'M+1 RETURN/VOLATILITY']].head()
```

[107]

	STOCK CODE	ISIN CODE	STOCK NAME	PREDICTION DATE	M+1 BOND PRICE	M+1 YIELD INCOME	M+1 ROLL DOWN RETURN	M0 EXPECTED CHG IN YIELD	M+1 BOND RETURN	M+1 BOND VOLATILITY	M+1 RETURN/VOLATILITY
22329	UI200090	MYBUI2000908	CIMB MTN 1826D 03.4.2025 - Issue No 8	2020-11-01	103.129337	51.564668	0.000061	0.000000	51.564730	0.071130	724.934082
22332	VE200066	MYBVE2000665	UEMS IMTN 3.70% 03.05.2021 - Issue No. 8	2020-11-01	100.196005	50.098002	-0.000070	0.006000	50.103933	0.078521	638.098457
22331	UI200083	MYBUI2000833	SCSB SENIOR CLASS A MTNS (SERIES 4)	2020-11-01	102.508600	51.254300	0.000201	0.000000	51.254501	0.231230	221.660191
22320	VI200059	MYBVI2000591	COUNTRY GDN IMTN 5.250% 27.03.2025- Issue No 7	2020-11-01	107.105719	53.552860	0.000259	0.060009	53.613128	0.242552	221.037836
22322	VK200019	MYBVK2000191	COUNTRY GDN IMTN 5.700% 02.03.2027 - Issue No 5	2020-11-01	110.923833	55.461917	0.000540	0.000000	55.462456	0.531685	104.314518

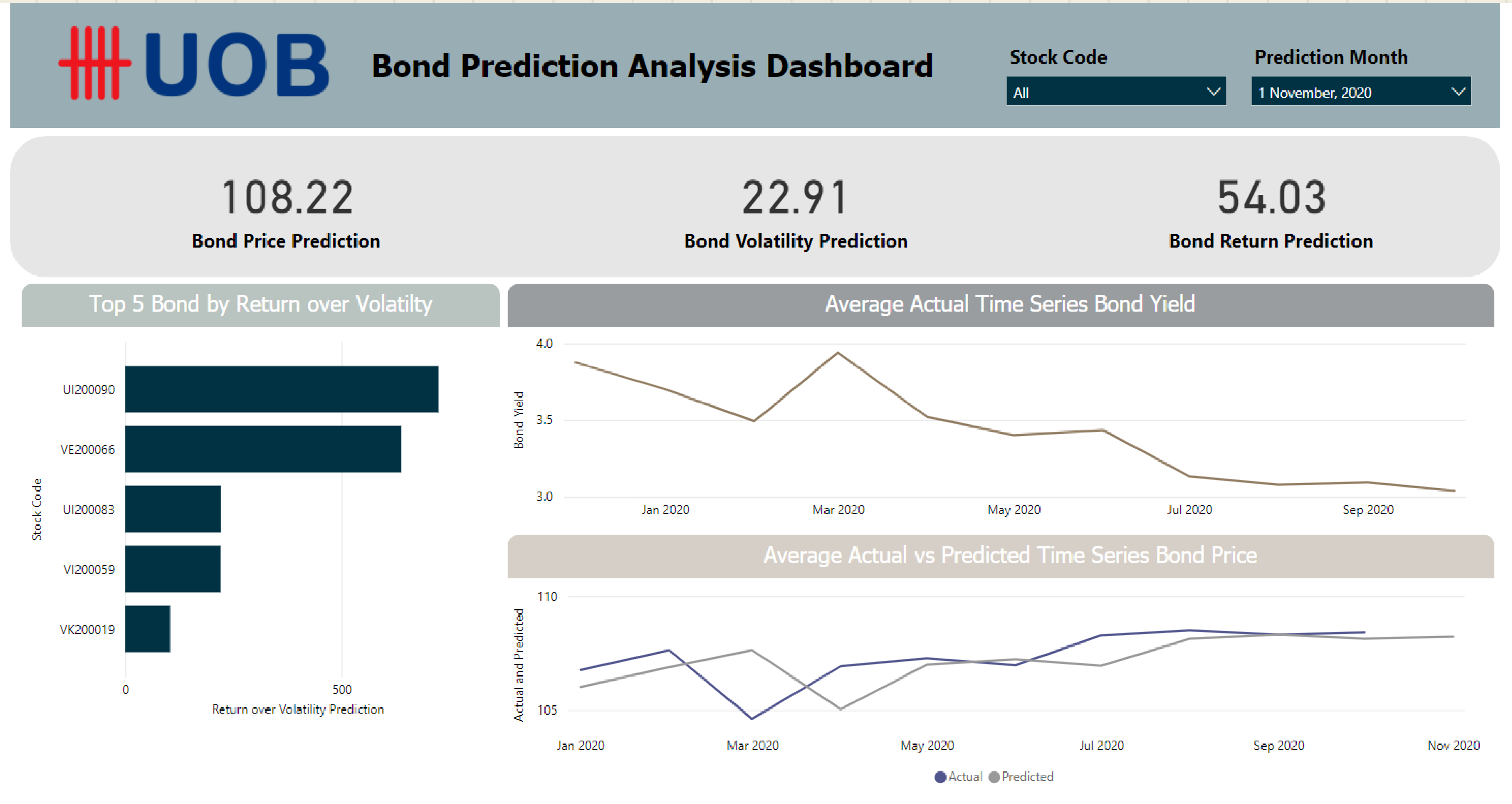
CUSTOMER SEGMENT

Bond manager

Challenges before making an investment decision:

- assess a large number of bond information
- constantly monitor the bond price movements
- check bond yields and return

DASHBOARD PROTOTYPE



WHY DOES OUR MODEL HAVE HIGHER FEASIBILITY OF SUCCESS AND POTENTIAL FOR FUTURE EXECUTION ?

1

High accuracy of approximately 99.65%

2

Used Correlation Matrix with Heatmap techniques to discover the features that have the biggest impact on the model

3

Can be trained easily and efficiently even on systems with relatively low computational power

4

Linear regression has a lower time complexity compared to other machine learning algorithms.

BUSINESS VALUE

1 Obtain accurate future bond price based on 15 months of large datasets

Accurate mean values with a smaller margin of error

2 Purchase bonds of highest return/volatility ratio based on our recommendation

Save time and reduce risks

3 Visualise the performance of each bonds monthly

See the difference of the actual bond price and predicted bond price

CONCLUSION

Our prototype is able to help our users predict bonds, calculate and to recommend them the bonds with the highest return/volatility ratio through machine learning.