Model_Run

February 16, 2019

1 This is Main file, containing all the work flow.

1.1 Python & SQL & Machine Learning Project

1.1.1 Use Decision Tree regression:

```
1. Select 10 stocks as independent variables to predict stocks return for the left 2990 stocks
2. Training sample should be at least 300 daily returns, and test sample at least 200 days.
3. Determine which stock has the smallest out-of-sample RMSE
4. Data Retrieval
 Get data from data source from yahoo
 Save data to the database
 Output result to DB
```

1.1.2 1. Import Packages

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.tree import DecisionTreeRegressor
        import math as mh
        import fix_yahoo_finance as yf
        import datetime
        import pyodbc
```

1.1.3 2. Read Russel 3000 tickers from Local file

```
In [ ]: # read tickers from csv
        symbol = pd.read_csv("symbol.csv")
        # reset column names
        symbol.columns = ['Company','Ticker']
        # get ticker list
        ticker = list(symbol['Ticker'])
```

1.1.4 3. Retrive data from yahoo finance

```
In [ ]: #retrive data for last two years from fix_yahoo_finance
        stocks = ticker
        start = datetime.date(2016,12,10)
        end = datetime.date(2018,12,10)
```

```
data = yf.download(stocks, start=start, end=end)
# get close price from dataframe 'data'
Close_data = data.Close
# drop data with NA value for the last two years
Close = Close_data.dropna(axis = 'columns')
```

1.1.5 4. Export user-input and original price data from python to database

```
In [ ]: from Model_Output import User_Output, Price_Output
        # set user-input data
        run_id = 1
        train_period = 300
        test_period = 200
        start_date = '2016-12-10'
        end_date = '2018-12-10'

        # export user-input data
        User_Output(run_id, train_period, test_period, start_date, end_date)
```

```
In [ ]: # export price data
        Price_Output(Close)
```

1.1.6 5. Import user-input data and price data from database to python

```
In [ ]: from Model_Input import Data_Input

        # set the test run_id as 1
        run_test = 1

        # Notes: Each run_id is unique. It contains all the unique information for this run.
        # This means we only need one parameter to get data we need.

        # Get one dataframe contaning price data from DB
        price_db = Data_Input(run_test)['df_price']
        # Get one dataframe contaning user-input data from DB
        # Data includes Start_Date, End_Date,Train_Period,Test_Period
        user_db = Data_Input(run_test)['df_dates']
```

1.1.7 6. Prepare data for regression training

```
In [ ]: # Convert DB dataframe to new dataframe where all columns' names are tickers
        # get all stock tickers
        price_db_tickers = list(set(price_db['Ticker']))
        # get all dates(use set to get unique value)
        price_db_dates = list(set(price_db['AsofDate']))
        # get the new Price dictionary where keys are tickers and values are their prices
        Db_data = {}
        for i in range(len(price_db_tickers)):
            Db_data[price_db_tickers[i]] = []
```

```
            for j in range(len(price_db_dates)):
                temp_p = price_db.loc[(price_db['Ticker'] == price_db_tickers[i]) &
                                      (price_db['AsofDate'] == price_db_dates[j])]
                Db_data[price_db_tickers[i]].append(temp_p['Price'])
        # transform dictionary to dataframe
        Db_data = pd.DataFrame(Db_data)
```

In [ ]:
```
        # Convert price data into log return data
        Log_ret =  np.log(Db_data / Db_data.shift(1))
        # drop na values in rows
        Ori_data = Log_ret.dropna(axis = 'rows')
        # Get all the ticker names
        col_names = list(Ori_data.columns)
        # Get the number of dates
        n_row = len(Ori_data)
        # Get all the dates and convert into string types
        Ori_date = list(Ori_data.index)
        for i in range(n_row):
            Ori_date[i] = str(Ori_date[i].date())
```

In [ ]:
```
        # get the selected prdictors from csv file
        f_symbol = pd.read_csv("SELECT.csv")
        features = list(f_symbol['TICKER'])
        # get the left dependent variables
        dependents = [i for i in col_names if i not in features]
        # get the number of features and dependent variables
        n_predicor = len(dependents)
```

1.1.8 7. Use Desicion Tree Regression model to train the data

In [8]:
```
        # define a function to calculate the RMSE for returns
        def calc_mse(l1,l2):
            length = len(l1)
            mse = 0
            for i in range(length):
                mse = mse + mh.pow((l1[i]-l2[i]),2)
            return(mh.sqrt(mse))
```

In [ ]:
```
        # train the model
        # set the predicted return as a dictionary(further into a dataframe)
        Ret_predict = {}
        # set the predicted return MSE as a list
        Ret_MSE = []
```

In [ ]:
```
        # get the number of training period
        t_p = user_db['Train_Period'][0]
        s_p = user_db['Test_Period'][0]
```

In [ ]:
```
        # use last day return to predict next day return using pre-selected stocks
        # seperate training set and testing set for independent variables
```

```
x_train = np.array(Ori_data[features][:t_p])
x_train = np.reshape(x_train,(t_p,len(features)))
x_test = np.array(Ori_data[features][t_p+1: t_p+s_p])
# using selected stocks to train each stock left
for i in range(n_predicor):
    # seperate training set and testing set for each dependent variables
    y_train = np.array(list(Ori_data[dependents[i]])[1:t_p + 1])
    y_test = np.array(list(Ori_data[dependents[i]])[t_p + 2 : t_p+s_p+1])
    # run regression tree model
    regr = DecisionTreeRegressor(max_depth=10)
    # model fitting
    regr.fit(x_train, y_train)
    # model prediction
    y_pre = regr.predict(x_test)
    # get dictionary for return prediction
    Ret_predict[dependents[i]] = y_pre
    # get the list for MSE value
    Ret_MSE.append(calc_mse(y_test,y_pre))
# get the dataframe for Return prediction
Ret_predict = pd.DataFrame(Ret_predict,index = Ori_data.index[t_p + 2 : t_p+s_p+1]
```

1.1.9 8. Export return prediction and mse value from python to database

In [ ]:
```
# Model result export
from Model_Output import Result_Output
Result_Output(run_id,features,Ret_predict,Ret_MSE)
```

Model_Output

February 16, 2019

1 This is Model Output file. From python to database.

```
In [ ]: def Result_Output(run_id,features,Ret_predict,Ret_MSE):
            import pyodbc
            conn=pyodbc.connect("Driver={SQL Server};" "Server=WIN-D32624325Z\SQLEXPRESS;"
                                "Database=master;" "Trusted_Connection=yes;")
            cursor=conn.cursor()

            n_features = len(features)

            # export Predictor
            run_id = [1] * n_features
            for i in range(n_features):
                cursor.execute("INSERT INTO Predictor([Run_id],[Ticker]) values (?,?)",
                               run_id[i],features[i])

            # export Pred_Ret
            run_id = [1] * len(Ret_predict) * len(Ret_predict.columns)
            for i in range(len(Ret_predict.columns)):
                for j in range(len(Ret_predict)):
                    cursor.execute("INSERT INTO Pred_Ret([Run_id],[Ticker],[AsofDate]"
                                   +",[Pre_ret]) values (?,?,?,?)",run_id[i+j]
                                   ,Ret_predict.columns[i],Ret_predict.index[j]"
                                   ,Ret_predict[Ret_predict.columns[i]][j])

            # export Pred_Mse
            run_id = [1] * len(Ret_predict.columns)
            for i in range(len(Ret_predict.columns)):
                cursor.execute("INSERT INTO Pred_Mse([Run_id],[Ticker],[Mse]) values (?,?,?)",
                               run_id[i],Ret_predict.columns[i],Ret_MSE[i])

            # export Run_result
            run_id = 1
            Min_mse = min(Ret_MSE)
            Min_ticker = Ret_predict.columns[Ret_MSE.index(Min_mse)]
            cursor.execute("INSERT INTO Run_result([Run_id],[Ticker],[Min_mse]) values (?,?,?)"
                           ,run_id,Min_ticker,Min_mse)
```

```
In [ ]: def User_Output(run_id, train_period, test_period, start_date, end_date):
            # Build connection with Database
            import pyodbc
            conn=pyodbc.connect("Driver={SQL Server};" "Server=WIN-D32624325Z\SQLEXPRESS;"
                                "Database=master;" "Trusted_Connection=yes;")
            cursor=conn.cursor()
            # Insert data into database
            cursor.execute("INSERT INTO Run([Run_id],[Train_Period],[Test_Period],"
                           + "[Start_Date],[End_Date]) values (?,?,?,?,?)"
                            ,run_id,train_period,test_period,start_date,end_date)

In [ ]: def Price_Output(close):
            # Build connection with Database
            import pyodbc
            conn=pyodbc.connect("Driver={SQL Server};" "Server=WIN-D32624325Z\SQLEXPRESS;"
                                "Database=master;" "Trusted_Connection=yes;")
            cursor=conn.cursor()
            # Insert price data into database
            for i in range(len(Close.columns)):
                for j in range(len(Close)):
                    cursor.execute("INSERT INTO Stock([Ticker],[AsofDate],[Price])"
                                   + "values (?,?,?)", Close.columns[i], Close.index[j]
                                   , Close[Close.columns[i]][j])
```

Model_Input

February 16, 2019

1 This is Model Input file. From database to python.

```
In [ ]: def Data_Input(run_id):

            import pandas as pd
            import pyodbc

            # connect database
            conn=pyodbc.connect("Driver={SQL Server};" "Server=WIN-D21B624326M\SQLEXPRESS;"
                                "Database=master;" "Trusted_Connection=yes;")

            # querry for Start_Date, End_Date,Train_Period,Test_Period from 'Run' table
            query_dates = "SELECT Start_Date, End_Date,Train_Period,Test_Period FROM Run "
            + "WHERE Run_id = '" + str(run_id) + "'"
            df_dates = pd.read_sql(query_dates,conn)

            # get price data from 'Stock' db
            query_price="SELECT * FROM Stock WHERE AsofDate BETWEEN '"
            + df_dates['Start_Date'][0] + "' AND '" + df_dates['End_Date'][0] + "'"
            df_price = pd.read_sql(query_price,conn)

            # return dictionary containing datae and price
            data = {}
            data['df_dates'] = df_dates
            data['df_price'] = df_price
            return(data)
```

Table design

We have 6 tables in total.

The first table is called Stock. We have three columns

Ticker (datatype varchar(255))

AsofDate (datatype Date)

Price (datatype float)

Column Name	Data Type	Allow Nulls
Ticker	nvarchar(MAX)	☐
AsofDate	date	☐
Price	float	☐

For this table, we have the data of 2782 stocks in 502 days so this table should have a total of 1396564 columns. Each column represents the price of a particular stock in a selected date. We treat Ticker and AsofDate as primary key to identify the required price.

The second table is called Run. We have five columns

Run_id (datatype int)

Train_Period (data type int)

Test_Period (data type int)

Start_Date (datatype Date)

End_Date (datatype Date)

Column Name	Data Type	Allow Nulls
Run_id	int	☐
Train_Period	int	☐
Test_Period	int	☐
Start_Date	date	☐
▶ End_Date	date	☐

For this table, we need a Run_id to indicate the results each time we run. Then we have four user inputs which tell us the starting date, ending date, which portion is the training period and the remaining is the testing period. We treat Run_id as primary key to identify the other four parameters.

The third table is called Predictor. We have two columns

Run_id (datatype int)

Ticker (datatype varchar(255))

Column Name	Data Type	Allow Nulls
Run_id	int	☐
Ticker	varchar(255)	☐

For this table, we store the ten tickers that we selected as independent variables. Similarly, we need a Run_id to indicate the results each time we run. We treat Run_id and Ticker as primary key. We need both of them because each time we run the Run_id

is the same for the ten selected stocks.

The fourth table is called Pred_Ret. We have four columns

Run_id (datatype int)
Ticker (datatype varchar(255))
AsofDate (datatype Date)
Pre_ret (datatype float)

Column Name	Data Type	Allow Nulls
Run_id	int	☐
Ticker	varchar(255)	☐
AsofDate	date	☐
Pred_ret	float	☐

For this table, we store the output. Similarly, we need a Run_id to indicate the results each time we run. This table shares the same structure as the first table(table Stock) except that we replace price with the calculated return. We treat Run_id, Ticker and AsofDate as primary key to identify the predicted return.

The fifth table is called Pred_Mse. We have three columns

Run_id (datatype int)
Ticker (datatype varchar(255))
Mse (datatype float)

Column Name	Data Type	Allow Nulls
Run_id	int	☐
Ticker	varchar(255)	☐
Mse	float	☐

For this table, we store the calculated RMSE. Similarly, we need a Run_id to indicate the results each time we run. Then we have ticker and its corresponding RMSE. For each ticker, we take the difference of actual return and predicted return each day and add up sums of all differences squared, the new take the square root to acquire RMSE. We treat Run_id and Ticker as primary key to identify RMSE.

The sixth table is called Run_Result. We have three columns

Run_id (datatype int)
Ticker (datatype varchar(255))
Min_mse (datatype float)

Column Name	Data Type	Allow Nulls
Run_id	int	☐
Ticker	varchar(255)	☐
Min_mse	float	☐

For this table, we store the ticker that has minimum RMSE. Similarly, we need a Run_id to indicate the results each time we run. For each Run_id, we would output name of the

ticker that has the lowest RMSE and its corresponding RMSE. We treat Run_id and Ticker as primary key to identify minimum RMSE.

MSSQL Codes

```
CREATE TABLE Stock (
	Ticker varchar(255) NOT NULL,
	AsofDate Date NOT NULL,
	Price float NOT NULL,
	PRIMARY KEY(Ticker,AsofDate)
);

CREATE TABLE Run (
	Run_id int NOT NULL,
	Train_Period int NOT NULL,
	Test_Period int NOT NULL,
	Start_Date Date NOT NULL,
	End_Date Date NOT NULL,
	PRIMARY KEY(Run_id)
);

CREATE TABLE Predictor (
	Run_id int NOT NULL,
	Ticker varchar(255) NOT NULL,
	PRIMARY KEY(Run_id,Ticker)
);

CREATE TABLE Pred_Ret (
	Run_id int NOT NULL,
	Ticker varchar(255) NOT NULL,
	AsofDate Date NOT NULL,
	Pre_ret float NOT NULL,
	PRIMARY KEY(Run_id,Ticker,AsofDate)
);

CREATE TABLE Pred_Mse (
	Run_id int NOT NULL,
	Ticker varchar(255) NOT NULL,
	Mse float NOT NULL,
	PRIMARY KEY(Run_id,Ticker)
);

CREATE TABLE Run_result (
	Run_id int NOT NULL,
	Ticker varchar(255) NOT NULL,
	Min_mse float NOT NULL,
	PRIMARY KEY(Run_id,Ticker) );
```