Topic: Mutual Fund Prediction and Suggestion:

Course: BE Project
Fr. Agnel Bandra

Members:
1) Manish Singh
2) Sunny Dodhiya

Overview and Motivation

Overview

- Predicting the best Mutual funds to invest in based on short term goals.

- The analysis was done on the historical returns data available. We selected top 10 Fund Families based on largest Asset Under Management (AUM). It had approximately 1277 funds from these 10 fund families comprising of all the different Morningstar categories like Large,MidCap,Small, Growth,Blend,Value,Index funds etc.

- The source of the data were from Morningstar.com, Yahoo finance. The data from Morningstar was scraped using Beautiful Soup and Pandas read_html libraries. Fund parameters like Alpha,Beta,Sharpe Ratio, Sortino Ratio,Standard Deviation, Returns information, Managemanet Information,Holdings information etc. was available in a annualized form on the website. The fund returns plots(funds comparison) were plotted using matplotlib libraries using Yahoo Finance API etc.

Motivation

- I (Manish) is a CFA Level 2 Candidate and thus have a good insight into the finance domain and have a experience of around 1 year in the industry as an intern.

- After discovering Machine Learning we were highly motivated to apply the algorithms we learnt in class on the real world data set.

Process

1. **Data :** Connecting to the Source, Scraping method, cleanup & Processing. Storing the Cleaned datasets in CSV.

2. **Regression Analysis :** We tried Linear Regression but Random Forest Classification outweighed it. Random Forest Classifer was used to predict Top performing funds with Accuracy of 92 percent.

3. **Visualisation :** Visualisations like scatter plot, box plot, bar plots

4. **Conclusion :** We found out Top performing fund for 3 years data available. In latest 3 years available data, the mutual fund with less Expense Ratio gives more returns on investments.

1) DATA : Pre Processing

1. **Web Scraping for fetching the Fund Information:** Here we are scraping the data from www.morningstar.com for approximately 1277 funds of 10 fund families. Hence the scraped data is exported in csv files.
2. **List of Top 10 Fund Families based on largest Asset Under Management (AUM):** This List includes Fund Families like Vanguard, American Funds, PIMCO, Fidelity Investments, Franklin Templeton Investments, BlacRock, T. Rowe Price, J.P.Morgan Funds, Oppenheimer Funds and Columbia

```
# Import Python Libraries
# special IPython command to prepare the notebook for matplotlib
%matplotlib inline
import requests
import numpy as np
import pandas as pd                                  # pandas
from io import StringIO
import matplotlib.pyplot as plt                      # module for plotting
import datetime as dt                                # module for manipulating dates and times
from collections import OrderedDict

# Import scipy library
import scipy as sp

# Import sklearn Libraries
import sklearn
import seaborn as sns

# Import module matlabplot for visaulizations
from matplotlib import pyplot as plt
from matplotlib import rcParams
import math

# Import Beautiful Soup library
import bs4
from bs4 import BeautifulSoup
import urllib
import urllib.request



# Defining a DataFrame for Fund Family which contains Fund Family and the Fund Family URL.
FFamily = pd.DataFrame(columns=['Fund_Family','MorningstarURL'])

# List of 10 largest Mutual Fund Families
FFamily.loc[0] = ['Vanguard','http://quicktake.morningstar.com/fundfamily/vanguard/0C00001YUF/fund-
list.aspx']
"""FFamily.loc[1] = ['American Funds','http://quicktake.morningstar.com/fundfamily/american-funds/
0C00001YPH/fund-list.aspx']
FFamily.loc[2] = ['PIMCO Funds','http://quicktake.morningstar.com/fundfamily/pimco/0C00004ALK/fund-
list.aspx']
FFamily.loc[3] = ['T. Rowe Price','http://quicktake.morningstar.com/fundfamily/t-rowe-price/
0C00001YZ8/fund-list.aspx']
FFamily.loc[4] = ['JP Morgan','http://quicktake.morningstar.com/fundfamily/jpmorgan/0C00001YRR/fund-
list.aspx']
FFamily.loc[5] = ['Fidelity Investments','http://quicktake.morningstar.com/fundfamily/fidelity-
investments/0C00001YR0/fund-list.aspx']
FFamily.loc[6] = ['Franklin Templeton Investments','http://quicktake.morningstar.com/fundfamily/
franklin-templeton-investments/0C00004AKN/fund-list.aspx']
FFamily.loc[7] = ['BlackRock','http://quicktake.morningstar.com/fundfamily/blackrock/0C000034YC/fund-
list.aspx']
FFamily.loc[8] = ['Columbia','http://quicktake.morningstar.com/fundfamily/columbia/0C00001YQG/fund-
list.aspx']
FFamily.loc[9] = ['Oppenheimer Funds','http://quicktake.morningstar.com/fundfamily/oppenheimerfunds/
0C00001YZF/fund-list.a
```

DATA: Scrapping and Cleaning

1. **Fund Family with Ticker :** E.g.Fidelity funds have ticker starting with F whereas Vanguard funds have ticker starting with V.List of Top 10 Fund Families based on largest Asset Under Management (AUM)
2. **Load Benchmarks for each Fund** :Each and every fund follows an index like the S&P 500 etc. which we call a benchmark, so that the fund's returns can be compared to the Index benchmark returns. Fetching the Fund's benchmark.

```
# Fund Family DataFrame
Funds_family = pd.DataFrame(columns=['Fund_Name','Fund_Family','Fund_Ticker'])
i = 0

for index in range(0,len(FFamily)):
    # For each fund, fetch the MorningStar URL
    contenturl = FFamily.MorningstarURL[index]
    # Using urllib library.
    page = urllib.request.urlopen(contenturl)
    # Using Beautiful Soup to read from page
    soup = BeautifulSoup(page)
    # Extract the information from the div which contains the class "syn_section_b1"
    table = soup.find("div", { "class" : "syn_section_b1" })
    # Loop to fetch all the fund tickers and its's names which is contained within href section of
the URL
    for row in table.findAll('a'):
        # If we carefully observe the URL, the ticker information starts at 73
        if (row['href'][73:]) != '':
            Funds_family.loc[i] = [row.contents[0],FFamily['Fund_Family'][index],row['href'][73:]]
            i = i+1
```

```
# Benchmark columns.
fund_benchmark_columns = ['Fund_Ticker','Benchmark_Index']
# Fund Benchmark DataFrame
fund_benchmark = pd.DataFrame(columns=fund_benchmark_columns)

# Loop for each fund for getting it's becnhmark
for i in range(0,len(Funds_family)):
    FUND_NAME = Funds_family['Fund_Ticker'][i]
    # try except exception block is used to ignore errors if any and moving forward for different
funds.
    try:
        # Below is the AJAX request URL whose table contains the Benchmark
        ratingriskurl = "http://performance.morningstar.com/RatingRisk/fund/mpt-statistics.action?
&t=XNAS:"+FUND_NAME+"&region=usa&culture=en-
US&cur=&ops=clear&s=0P00001MK8&y=3&ep=true&comparisonRemove=null&benchmarkSecId=&benchmarktype="
        # Read the 0th value of the array
        mpt_statistics_bench = pd.read_html(ratingriskurl)[0]
        # Filtering all the not null values
        mpt_statistics_bench = mpt_statistics_bench[mpt_statistics_bench.Alpha.notnull()]
        # After filtering the row at index at 1 has the bechmark
        mpt_statistics_bench = mpt_statistics_bench.reset_index(drop=True).iloc[[1]]
        mpt_statistics_bench = mpt_statistics_bench.ix[:,0:2]
        mpt_statistics_bench.columns = fund_benchmark_columns
        # Append fund benchmark dataframe for each fund
        fund_benchmark = fund_benchmark.append(mpt_statistics_bench)
    except:
        # Those Funds which have error in finding benchmark are printed.
        print("Index : ",i,"No Benchmark Data Found in Morningstar for Fund : ",FUND_NAME)

# Reset the fund_benchmark dataframe.
fund_benchmark = fund_benchmark.reset_index(drop=True)
```

DATA: Exporting Processed Data to CSV

Tweaks and Adjustment : Exporting the processed data as data frame for further Analysis into a Defined Format, as CSV

```
# Initializing the Fund Returns object.
fund_returns = {}

# 15 year returns data for each fund.
for i in range(0,len(fund_df)):
    # Fetch the Ticker.
    fund_name = fund_df['Fund_Ticker'][i]
    # Fetch the Benchmark
    fund_benchmark_var = fund_df['Benchmark_Index'][i]
    # Key to be stored
    key = "fund_returns_"+fund_name
    # Fetching the Fund's benchmark and it s symbol
    symbol = (mstar_benchmark_symbol[mstar_benchmark_symbol['Benchmark_Index'] ==
fund_benchmark_var]['Mstar_Symbol']).values[0]
    # Fund Returns URL, passing the Fund Ticker, the bechmark symbol and 15 years of
data to the URL
    fund_ret_url = "http://performance.morningstar.com/Performance/fund/performance-
history-1.action?&t=XNAS:"+fund_name+"&region=usa&culture=en-
US&cur=&ops=clear&s="+symbol+"&ndec=2&ep=true&align=m&y=15&comparisonRemove=false&locca
t=&taxadj=&benchmarkSecId=&benchmarktype="
    # Try Except block in case to encounter any errors
    try:
        # fetching and storing data

        fund_history = get_performance_data(fund_ret_url)


        #fund_history = fund_history.reset_index()
    except:
        # No returns Information then print and store an empty array in the value for
that fund.
        print("No Returns Data available for Fund : ",fund_name)
        # Creating an Empty DataFrame for fund with no returns
        fund_history = pd.DataFrame()
    # Assigning the DataFrame to the value
    fund_returns[key] = fund_history



# Export to csv files:

for i in range(0, len(fund_returns)):
    fund_key = list(fund_returns)[i]
    # Fund Returns filename
    fund_filename = list(fund_returns)[i][-5:]+"_RETURNS.csv"
    # Storing the Fund Returns values in a csv
    fund_returns[fund_key].replace(to_replace=u'\u2019',value='').to_csv(fund_filename)
```

2) Regression Analysis

We read the fund data from CSV files saved earlier via Scraping.
We use data analysis and regression, Random Forest to predict Top performing funds which are good for investment.
We have collected past 10 years data of over 1000 funds. We will analyze the value of alpha, beta, return, R squared, sharpe ratio, standard deviation, expense ratio for each fund for 3 years and fit a model to predict top performing funds

Read Fund Meta data from CSV

Read fund meta data from csv file. This includes fund ticker, fund family and benchmark. We will use fund ticker later to fetch returns and statistics for each fund from csv file.

```
# get fund meta data from CSV
fund_meta_data = pd.read_csv("../Fund_Data/Fund_Metadata.csv").drop(u'Unnamed: 0',axis =1)
fund_meta_data.head()
```

Out[6]:

	Fund_Name	Fund_Family	Fund_Ticker	Benchmark_Index
0	Vanguard 500 Index Inv	Vanguard	VFINX	S&P 500 TR USD
1	Vanguard Balanced Index Inv	Vanguard	VBINX	Morningstar Moderate Target Risk
2	Vanguard CA Interm-Term Tax-Exempt Inv	Vanguard	VCAIX	Barclays Municipal TR USD
3	Vanguard CA Long-Term Tax-Exempt Inv			

Predict Good Fund for 3 years

- **Approach / Criteria** :

1. A good fund should give maximum returns with lesser risk i,e, it should be less volatile. A fund may be giving > 30 % returns based on its past performance but if it has a high volatility/high risk and we tend to loose more in case of loss. We used the information avaialble on Investopedia to determine what should be the parameter values for a good fund.

2. Expense ratio is the price paid by investor. It plays an important role while deciding fund's overall returns. e.g. A fund is giving 11% returns with expense ratio of 1% is better than a fund giving 13% Returns with Expense ratio of 6%. So we will calculate net returns as difference between returns and the Expense Ratio.

3. A good fund should have the following values :
   - R squared : between 85-100.
   - Return : as high as possible (we consider return > 10)
   - Beta : greater than 1
   - alpha : greater than 0
   - sharpe ratio : greater than 0
   - Standard deviation : low (we considered std deviation <15)

Creating the Decision Matrix

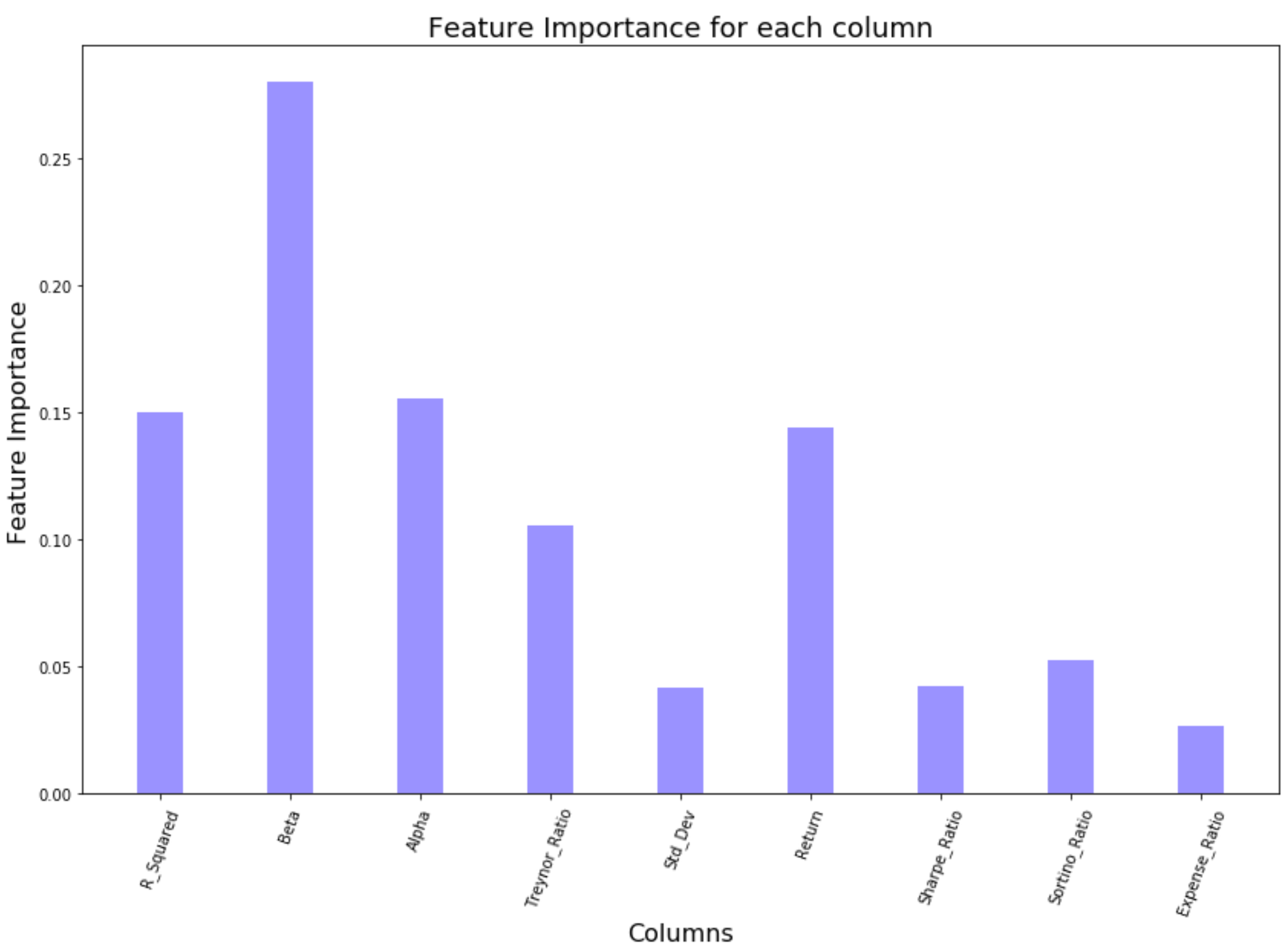
- **Matrix X :** Create X from the available data. Consider X as the alpha, beta, sharpe ratio, returns, standard deviation, rsquared,etc parameters.

- **Matrix Y :** Create a Y array which gives 1 and 0 values for each fund based on parameters such as R squared value, alpha, beta, standard deviation, sharpe ratio, returns.
  Y as good_or_bad indicator which we calculated above.

Random Forest Classifier and F1 Scoring

We use Random forest classifier for number of trees ranging from 1 to 41 to find out which trees gives more accuracy. We will use f1 scoring parameter for cross validation.

Based on best Accuracy we select the most Appropriate n-estimator.



Best Features with high Correlation are R_Squared, Beta, Alpha and Returns.

Prediction

Now predict good/bad funds based on random forest model fitted for X and Y.

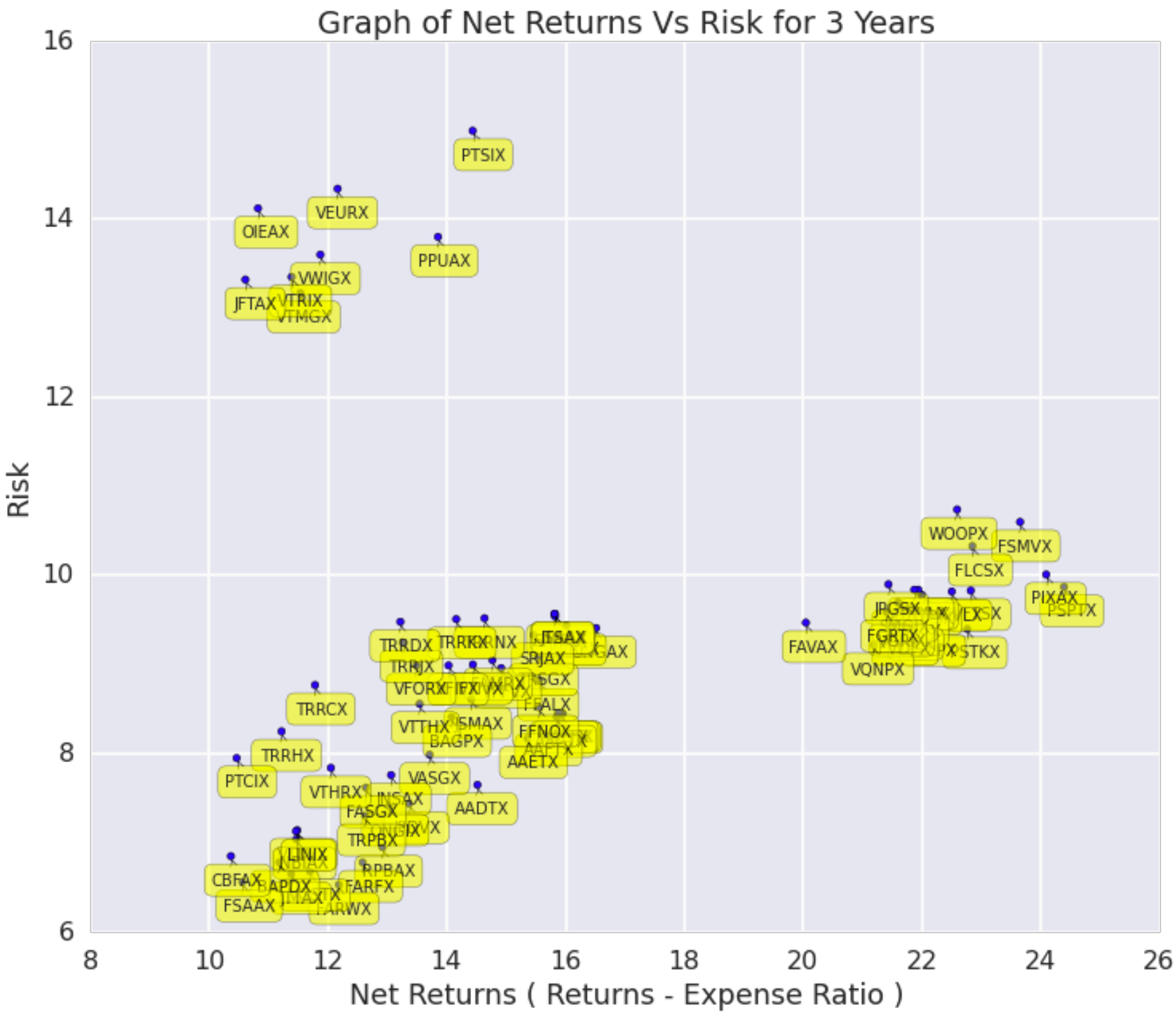
```
# Predict the result
Y_predict = clf.predict(X)
fund_data['Predicted_result'] = Y_predict

# sort the result in descending order of returns
good_fund_three_yrs =
fund_data.query("Predicted_result==1").sort(['Net_returns'],ascending=False).reset_index(drop=True)
```

Thus we achieve top predicted funds in sorted order. We Consider the top 20 and analyse them.

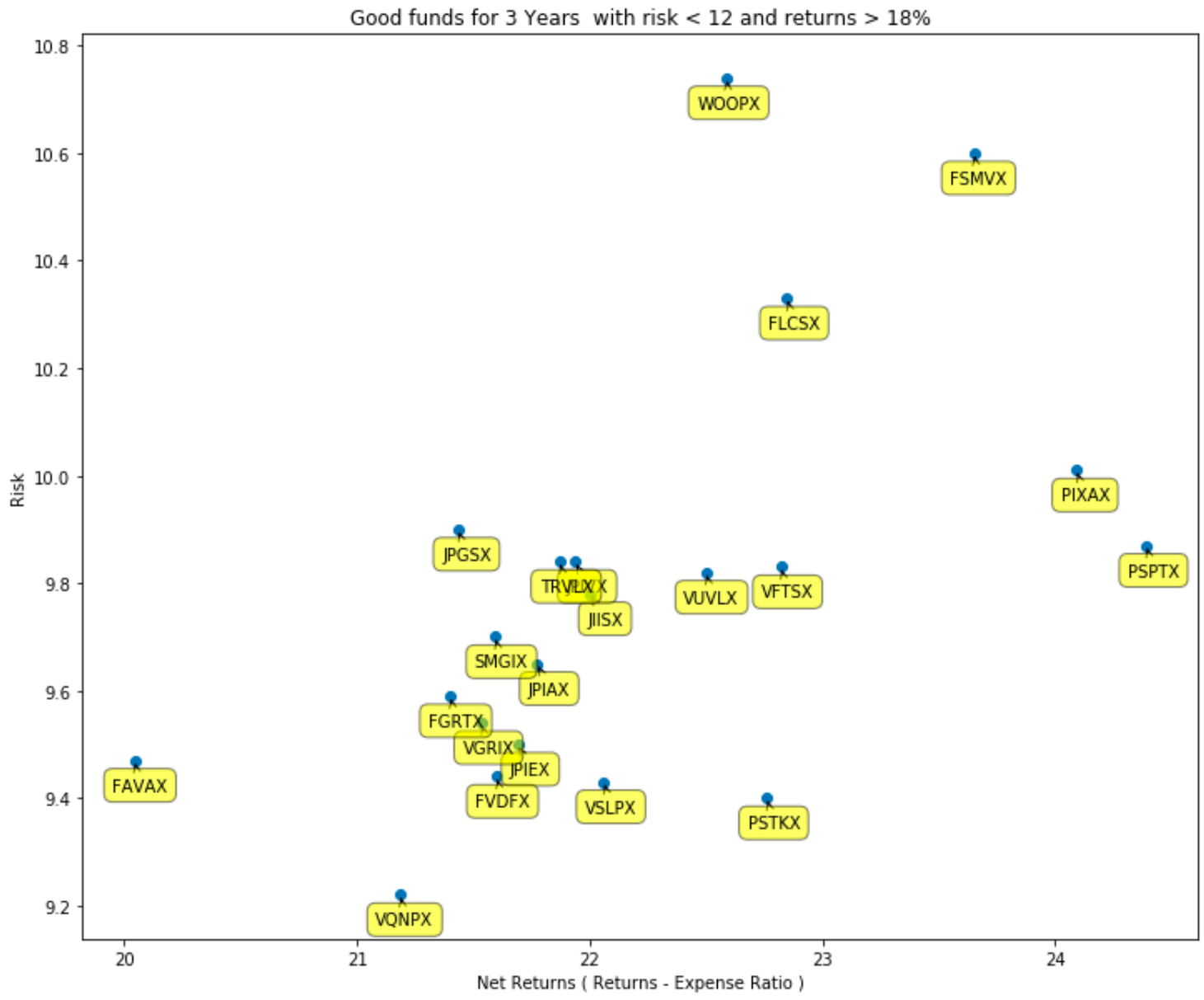
3) Visualisation

- We can see that there’s a negative correlation between Expense Ratio and Returns

Risk vs Return Plot



Good funds with risk < 12 and returns > 18 %



Summary and Conclusion

Summary for 3 years investment.

We used Random Foreset classifer to predict top performing funds for 3 year investment. We used cross validtaion to find the best value for n_estimator and predicted top 20 funds for investment. Based on our analysis PSPTX, PIXAX ,FSMVX, FLCSX, VFTSX are one of the top performing funds for 3 year investments which give more tha 23% returns with less tha 11% risk. Also fund with less expense ratio give more returns for 3 year investment.