# DS-520 Data Analysis and Decision Model

## FINAL PROJECT

### Presented By

* Mir Ahmed, DS-520 WEB-NM-21FATR
* Joel Baruch, DS-520 PHYB4-21FATR
* Nilanjana Chatterjee, DS-520 PHYB4-21FATR
* Weiwan Zhao, DS-520 PHYB4-21

# Table of contents

#### 1) Introduction

#### 2) Executive Summary

#### 3) Part 1: Data Collection

#### 4) Part 2: Exploratory Data Analysis

#### 5) Part 3: Hypothesis Testing

#### 6) Part 4: Linear Regression Model and Anova

# Introduction

A mutual fund is a company that pools money from many investors and invests the money in securities such as stocks, bonds, and short-term debt. The combined holdings of the mutual fund are known as its portfolio. The level of risk in a mutual fund depends on what it invests in. Usually, the higher the potential returns, the higher the risk will be. For example, stocks are generally riskier than bonds, so an equity fund tends to be riskier than a fixed income fund.

Some specialty mutual funds focus on certain kinds of investments, such as emerging markets, to try to earn a higher return. These kinds of funds also tend to have a greater risk of a larger drop in value. Given the general belief among people that mutual funds tend to be risker and are not safe to invest. Following is your null hypothesis.

H0 : Null Hypothesis: Mutual funds are risky and does not give much returns.

Average rate of return (µ) <= ~2 %

Alternate Hypothesis you are trying to prove if Mutual funds are safe to invest.

Ha : Alternate Hypothesis : Mutual funds are safe and give good returns.

Average rate of return (µ) > 2%

# Executive Summary

We collected our datasets from gthe following URLs using web scraping:

1. Funds' Overview: <https://fundresearch.fidelity.com/fund-screener/results/table/overview/averageAnnualReturnsYear3/desc/>
2. Funds' Risks: <https://fundresearch.fidelity.com/fund-screener/results/table/risk/averageAnnualReturnsYear3/desc/>
3. Funds' Yields: <https://fundresearch.fidelity.com/fund-screener/results/table/daily-pricing-yields/averageAnnualReturnsYear3/desc/>

After collecting the datasets, we cleaned them and merged them to create our final dataset which is used for exploratory analysis, hypothesis testing, linear regression model and ANOVA regression analysis.

Through hypthesis testing, we found that the mutual funds' returns in USA are way above 2% in general (rejecting uor null hypothesis).

We were able to create a linear regression model which gave 94.45% accuracy level for prediction.

Then through the ANOVA table we were able gto conclude each of the explanatory variable (features) picked for our model is statistically significant

# Part 1: Collect Your Dataset

For this part, we chose to collect our data from <https://fundresearch.fidelity.com/fund-screener/>, which has data on 9626 funds - the comprehensive set of Mutual Funds in the USA.

We collected the data in 2 phases:

First we collected scraped the for overview and risk sepearately.

Then we cleaned and merged them to create our final dataset.

#dependencies and setup  
import pandas as pd  
import requests  
from bs4 import BeautifulSoup  
from splinter import Browser  
from webdriver\_manager.chrome import ChromeDriverManager  
import time  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import r2\_score  
import seaborn as sns  
import scipy.stats as stats  
import statsmodels.api as sm

# create a browser instance using splinter  
executable\_path = {'executable\_path': ChromeDriverManager().install()}  
browser = Browser('chrome', \*\*executable\_path, headless=False)  
time.sleep(1)

### Scraping overview data

# Empty lists  
names = []  
ms\_cat = []  
ytd\_daily = []  
yr1 = []  
yr3= []  
yr5 = []  
yr10 = []  
life\_of\_fund = []  
net\_expense\_ratio = []  
gross\_expense\_ratio = []  
ms\_rating\_overall = []  
  
for i in range(1,98):  
 # visit fidelity URL  
 fidelity\_url = f"https://fundresearch.fidelity.com/fund-screener/results/\  
 table/overview/averageAnnualReturnsYear3/desc/{i}?assetClass=&category=&order=assetClass%2Ccategory"  
 browser.visit(fidelity\_url)  
 time.sleep(4)  
  
 # create HTML object  
 html = browser.html  
  
 # parse HTML with BeautifulSoup  
 soup = BeautifulSoup(html, 'html.parser')  
  
 div = soup.find('div', id ='static-table-container')  
 table = div.find('table', id = 'static-table')  
 tbody = table.find('tbody', id = 'static-tbody')  
 for listing in tbody.find\_all('td', class\_ = 'name left'):  
 for name in listing.find\_all('a'):  
 names.append(name.text)  
 div2 = soup.find('div', id ='scrollable-results-table-wrapper')  
 table2 = div2.find('table', id = 'scrollable-results-table')  
 tbody2 = table2.find('tbody', id = 'results-tbody')  
 for listing in tbody2.find\_all('td', class\_ = 'morningstarCategory left'):  
 ms\_cat.append(listing.text)  
 for listing in tbody2.find\_all('td', class\_ = 'ytdDaily right'):  
 ytd\_daily.append(listing.text)  
 for listing in tbody2.find\_all('td', class\_ = 'yr1 left'):  
 yr1.append(listing.text)  
 for listing in tbody2.find\_all('td', class\_ = 'yr3 left sorted-column-cell'):  
 yr3.append(listing.text)  
 for listing in tbody2.find\_all('td', class\_ = 'yr5 left'):  
 yr5.append(listing.text)  
 for listing in tbody2.find\_all('td', class\_ = 'yr10 left'):  
 yr10.append(listing.text)  
 for listing in tbody2.find\_all('td', class\_ = 'lifeOfFund left'):  
 life\_of\_fund.append(listing.text)  
 for listing in tbody2.find\_all('td', class\_ = 'netExpenseRatio right'):  
 net\_expense\_ratio.append(listing.text)  
 for listing in tbody2.find\_all('td', class\_ = 'grossExpenseRatio right'):  
 gross\_expense\_ratio.append(listing.text)  
 for listing in tbody2.find\_all('td', class\_ = 'morningstarRatingOverall center'):  
 if type(listing.find('span')) == type(None):  
 ms\_rating\_overall.append("")  
 else:  
 ms\_rating\_overall.append(listing.find('span').text)  
 print(f"{len(names)} funds scraped until page {i}")

# creating overview dataframe  
df\_overview = pd.DataFrame({'name' : names, 'morningstar\_category': ms\_cat, 'ytdDaily': ytd\_daily, 'yr1': yr1,\  
 'yr3': yr3, 'yr5': yr5, 'yr10': yr10, 'life\_of\_fund': life\_of\_fund,\  
 'net\_expense\_ratio': net\_expense\_ratio, 'gross\_expense\_ratio': gross\_expense\_ratio,\  
 'morningstar\_rating\_overall': ms\_rating\_overall})

# cleaning overview dataframe  
for col in df\_overview.columns[2:-1]:  
 df\_overview[f"{col}"] = df\_overview[f"{col}"].str.replace('%', '', regex=True)  
 df\_overview[f"{col}"] = df\_overview[f"{col}"].str.replace('+', '', regex=True)  
 df\_overview[f"{col}"] = pd.to\_numeric(df\_overview[f"{col}"], errors='coerce')

### Scraping risk data

names = []  
categories = []  
risks = []  
stds = []  
srs = []  
betas = []  
r2s = []  
  
for i in range(1,98):  
 fidelity\_url = f"https://fundresearch.fidelity.com/fund-screener/results/\  
 table/risk/averageAnnualReturnsYear3/desc/{i}?assetClass=&category=&order=assetClass%2Ccategory"  
 browser.visit(fidelity\_url)  
 time.sleep(4)  
  
 # create HTML object  
 html = browser.html  
  
 # parse HTML with BeautifulSoup  
 soup = BeautifulSoup(html, 'html.parser')  
  
 div = soup.find('div', id ='static-table-container')  
 table = div.find('table', id = 'static-table')  
 tbody = table.find('tbody', id = 'static-tbody')  
 for listing in tbody.find\_all('td', class\_ = 'name left'):  
 for name in listing.find\_all('a'):  
 names.append(name.text)  
  
 div2 = soup.find('div', id ='scrollable-results-table-wrapper')  
 table2 = div2.find('table', id = 'scrollable-results-table')  
 tbody2 = table2.find('tbody', id = 'results-tbody')  
 for listing in tbody2.find\_all('td', class\_ = "morningstarCategory left"):  
 category = listing.find('span').text  
 categories.append(category)  
 for listing in tbody2.find\_all('td', class\_ = "morningstarCategoryRisk center"):  
 risk = listing.find('div', class\_ = "risk-icon-gradient")  
 risks.append(risk.get("class", "")[2][-1])  
 for listing in tbody2.find\_all('td', class\_ = "standardDeviation right"):  
 if type(listing.find('span')) == type(None):  
 stds.append("")  
 else:  
 stds.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "sharpeRatio3Yr right"):  
 if type(listing.find('span')) == type(None):  
 srs.append("")  
 else:  
 srs.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "beta right"):  
 if type(listing.find('span')) == type(None):  
 betas.append("")  
 else:  
 betas.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "r2 right"):  
 if type(listing.find('span')) == type(None):  
 r2s.append("")  
 else:  
 r2s.append(listing.find('span').text)  
 print(f"{len(names)} funds scraped until page {i}")

# creating risk dataframe  
df\_risk = pd.DataFrame({'name' : names, 'morningstar\_category': categories, 'risk': risks, 'std\_dev': stds,\  
 'sharpe\_ratio\_3\_yr': srs, 'beta': betas, 'r2': r2s})

### Scraping yield data

names = []  
morningstar\_category = []  
nav\_dollar\_amount = []  
nav\_change\_dollar\_amount = []  
nav\_change\_pct = []  
daily\_30\_day\_yield = []  
daily\_7\_day\_yield = []  
minimum\_investment = []  
last\_dividend = []  
morningstar\_rating\_overall = []  
  
for i in range(1,98):  
 fidelity\_url = f"https://fundresearch.fidelity.com/fund-screener/results/\  
 table/daily-pricing-yields/averageAnnualReturnsYear3/desc/{i}?assetClass=&category=&order=assetClass%2Ccategory"  
 browser.visit(fidelity\_url)  
 time.sleep(4)  
  
 # create HTML object  
 html = browser.html  
  
 # parse HTML with BeautifulSoup  
 soup = BeautifulSoup(html, 'html.parser')  
  
 div = soup.find('div', id ='static-table-container')  
 table = div.find('table', id = 'static-table')  
 tbody = table.find('tbody', id = 'static-tbody')  
 for listing in tbody.find\_all('td', class\_ = 'name left'):  
 for name in listing.find\_all('a'):  
 names.append(name.text)  
  
 div2 = soup.find('div', id ='scrollable-results-table-wrapper')  
 table2 = div2.find('table', id = 'scrollable-results-table')  
 tbody2 = table2.find('tbody', id = 'results-tbody')  
 for listing in tbody2.find\_all('td', class\_ = "morningstarCategory left"):  
 if type(listing.find('span')) == type(None):  
 morningstar\_category.append("")  
 else:  
 morningstar\_category.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "navDollarAmount right"):  
 if type(listing.find('span')) == type(None):  
 nav\_dollar\_amount.append("")  
 else:  
 nav\_dollar\_amount.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "navChangeDollarAmount right"):  
 if type(listing.find('span')) == type(None):  
 nav\_change\_dollar\_amount.append("")  
 else:  
 nav\_change\_dollar\_amount.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "navChangePct right"):  
 if type(listing.find('span')) == type(None):  
 nav\_change\_pct.append("")  
 else:  
 nav\_change\_pct.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "daily30DayYield right"):  
 if type(listing.find('span')) == type(None):  
 daily\_30\_day\_yield.append("")  
 else:  
 daily\_30\_day\_yield.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "daily7DayYield right"):  
 if type(listing.find('span')) == type(None):  
 daily\_7\_day\_yield.append("")  
 else:  
 daily\_7\_day\_yield.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "minimumInvestment right"):  
 if type(listing.find('span')) == type(None):  
 minimum\_investment.append("")  
 else:  
 minimum\_investment.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "lastDividend right"):  
 if type(listing.find('span')) == type(None):  
 last\_dividend.append("")  
 else:  
 last\_dividend.append(listing.find('span').text)  
 for listing in tbody2.find\_all('td', class\_ = "morningstarRatingOverall center"):  
 if type(listing.find('span')) == type(None):  
 morningstar\_rating\_overall.append("")  
 else:  
 morningstar\_rating\_overall.append(listing.find('span').text)  
 print(f"{len(names)} funds scraped until page {i}")

# creating yield dataframe  
df\_yield = pd.DataFrame({'name' : names, 'morningstar\_category': morningstar\_category, 'nav\_dollar\_amount': nav\_dollar\_amount,\  
 'nav\_change\_dollar\_amount': nav\_change\_dollar\_amount, 'daily\_30\_day\_yield': daily\_30\_day\_yield,\  
 'daily\_7\_day\_yield': daily\_7\_day\_yield, 'minimum\_investment': minimum\_investment,\  
 'last\_dividend':last\_dividend, 'morningstar\_rating\_overall': morningstar\_rating\_overall})

# cleaning yield dataframe  
df\_yield.drop(columns=['morningstar\_category','nav\_dollar\_amount', 'nav\_change\_dollar\_amount'\  
 , 'daily\_30\_day\_yield', 'daily\_7\_day\_yield', 'morningstar\_rating\_overall'], inplace = True)  
for col in df\_yield.columns[1:]:  
 df\_yield[f"{col}"] = df\_yield[f"{col}"].str.replace('$', '', regex=True)  
 df\_yield[f"{col}"] = df\_yield[f"{col}"].str.replace(',', '', regex=True)  
 df\_yield[f"{col}"] = pd.to\_numeric(df\_yield[f"{col}"], errors='coerce')

# close the browser session   
browser.quit()

### Merging the dataframes

# merging overview with risk  
df = pd.merge(df\_overview, df\_risk, on = "name", how = "left")  
del df['morningstar\_category\_y']  
df.rename(columns={"morningstar\_category\_x": "morningstar\_category"}, inplace = True)

# merging yield  
df = pd.merge(df, df\_yield, on = "name", how = "left")

# Exporting the dataframe into a csv file  
df.to\_csv('fidelity\_mutual\_funds\_return\_w\_risk.csv', index = False)

# Part 2: Exploratory Data Analysis

## Understanding the data

fidelity\_df = pd.read\_csv('fidelity\_mutual\_funds\_return\_w\_risk.csv')   
fidelity\_df.head()

name morningstar\_category \  
0 Baron Partners Fund Institutional Shares (BPTIX) Large Growth   
1 Baron Partners Fund Retail Shares (BPTRX) Large Growth   
2 Morgan Stanley Institutional Fund, Inc. Incept... Small Growth   
3 Morgan Stanley Institutional Fund, Inc. Incept... Small Growth   
4 Morgan Stanley Institutional Fund, Inc. Incept... Small Growth   
  
 ytdDaily yr1 yr3 yr5 yr10 life\_of\_fund net\_expense\_ratio \  
0 44.60 110.27 65.56 47.54 29.02 27.02 1.30   
1 44.28 109.72 65.13 47.15 28.68 20.45 1.56   
2 20.05 81.15 55.11 38.37 22.37 14.14 1.00   
3 19.75 80.67 54.70 37.98 22.01 13.83 1.35   
4 NaN 79.39 53.46 36.90 21.11 13.00 2.10   
  
 gross\_expense\_ratio morningstar\_rating\_overall risk std\_dev \  
0 1.30 1137.0 6 40.41   
1 1.56 1137.0 6 40.39   
2 1.19 574.0 7 40.44   
3 1.45 574.0 7 40.48   
4 2.27 574.0 7 40.42   
  
 sharpe\_ratio\_3\_yr beta r2 minimum\_investment last\_dividend   
0 1.60 1.51 0.63 1000000.0 0.2224   
1 1.59 1.51 0.63 2500.0 0.1243   
2 1.34 1.40 0.71 5000000.0 0.0000   
3 1.32 1.40 0.71 2500.0 0.0000   
4 1.30 1.40 0.71 2500.0 NaN

# counting unique values  
unique\_fund\_names = len(pd.unique(fidelity\_df['name']))  
total\_fund\_names = len(fidelity\_df)  
  
print(f'there are {unique\_fund\_names} \  
 funds in this data set of {total\_fund\_names} funds')

there are 9626 funds in this data set of 9626 funds

We can see that there are no duplicate samples in the dataset. Let's further explore the distributions and qualities of our data:

fidelity\_df.describe()

ytdDaily yr1 yr3 yr5 yr10 \  
count 9359.000000 9520.000000 9204.000000 8922.000000 7283.000000   
mean 12.841688 25.532315 12.170789 9.872851 8.455235   
std 12.137653 20.381654 8.276280 7.080595 5.485703   
min -22.090000 -37.660000 -29.380000 -23.430000 -19.020000   
25% 1.890000 6.010000 5.460000 3.740000 3.760000   
50% 11.970000 26.135000 11.265000 9.120000 8.180000   
75% 22.070000 40.180000 16.950000 14.160000 12.530000   
max 83.850000 146.460000 65.560000 47.540000 29.020000   
  
 life\_of\_fund net\_expense\_ratio gross\_expense\_ratio \  
count 9613.000000 9625.000000 9624.000000   
mean 7.796146 1.154081 1.412057   
std 5.245337 0.570965 1.333127   
min -11.740000 0.000000 0.000000   
25% 4.650000 0.760000 0.850000   
50% 7.080000 1.050000 1.190000   
75% 10.170000 1.490000 1.700000   
max 82.450000 5.250000 46.990000   
  
 morningstar\_rating\_overall risk std\_dev \  
count 9102.000000 9626.000000 9154.000000   
mean 464.152604 5.271556 14.547154   
std 362.456368 1.665109 8.134948   
min 10.000000 0.000000 0.180000   
25% 186.000000 4.000000 7.192500   
50% 361.000000 6.000000 16.080000   
75% 631.000000 6.000000 19.870000   
max 1250.000000 9.000000 59.740000   
  
 sharpe\_ratio\_3\_yr beta r2 minimum\_investment \  
count 9154.000000 8189.000000 8189.000000 9.624000e+03   
mean 0.784549 0.788115 0.805881 2.455934e+05   
std 0.391578 1.218248 0.282312 8.381462e+05   
min -2.450000 -21.000000 0.000000 0.000000e+00   
25% 0.550000 0.820000 0.810000 2.500000e+03   
50% 0.780000 0.960000 0.930000 2.500000e+03   
75% 1.030000 1.050000 0.970000 5.000000e+03   
max 2.610000 8.750000 1.000000 1.000000e+07   
  
 last\_dividend   
count 9178.000000   
mean 0.116582   
std 0.499280   
min 0.000000   
25% 0.013883   
50% 0.034480   
75% 0.118995   
max 24.012771

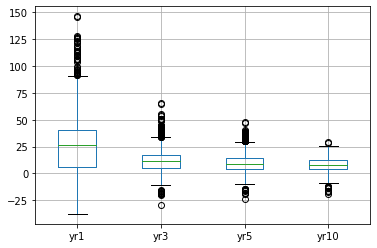
Seeing the description of the data, and focusing on the one, three, five and ten year returns, we do see that the means are always positive and above 7.7%. We also see that the standard deviation in each of our samplingn columns gets smaller over the years, which makes sense as funds' returns would "smoothe" over time, creating smaller and smaller deviations.

Beyond that, we do see that for each investment horizon (1, 3, 5, and 10 years) there are those funds that lose money, but we also see that they are a minority, as the 25th percentile is positive in all horizons.

Let's investigate further with some box plots:

fidelity\_df.boxplot(['yr1', 'yr3', 'yr5', 'yr10'])

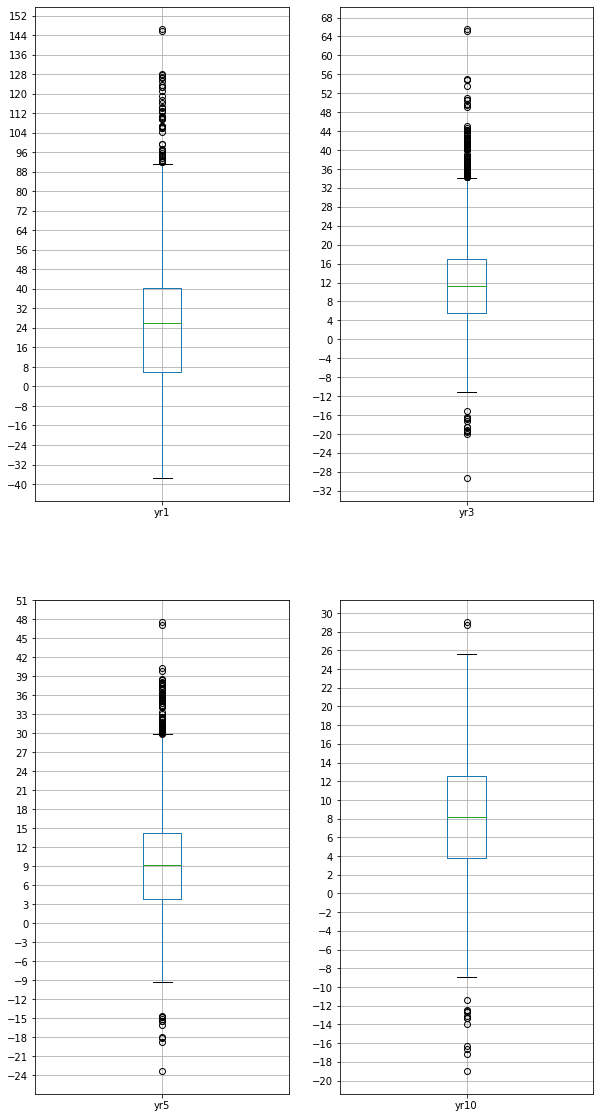
<AxesSubplot:>



Again, seeing the bulk of the performances in the positive ranges. Also, we see how the yr1 spread is wider and gets smaller and smaller for the following time horizons. Still, this clouds the visibility into the other horizons.

Let's break up the plots and customize the size and granularity of the grids:

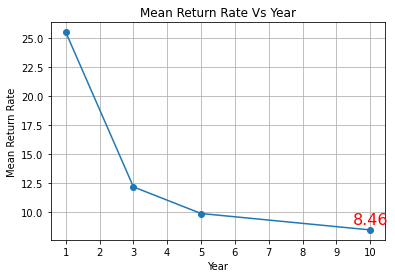
fig, ax = plt.subplots(2,2, figsize=(10,20))  
  
for idx, column in enumerate(fidelity\_df.columns[3:7]):  
 ax.flat[idx].yaxis.set\_major\_locator(plt.MaxNLocator(30))  
 fidelity\_df.boxplot([column], ax=ax.flatten()[idx])



Still, we see that the bulk of the measurments for all time horizons is safely above the 2% mark. We're begining to be optimistic about the average performance of an arbitrary mutual fund.

Furhter, something we'd like to point out and visualize, is how the average return does trend down over the time horizons:

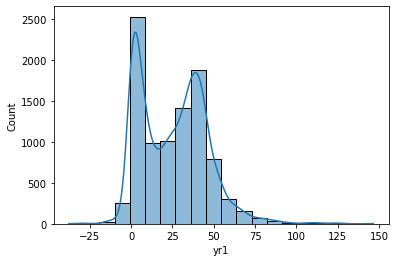
year = [1,3,5,10]  
mean\_return = [fidelity\_df['yr1'].mean(), fidelity\_df['yr3'].mean() ,   
 fidelity\_df['yr5'].mean(), fidelity\_df['yr10'].mean()]  
   
plt.plot(year, mean\_return, marker = 'o')  
plt.grid(zorder=0)  
plt.title('Mean Return Rate Vs Year')  
plt.xlabel('Year')  
plt.ylabel('Mean Return Rate')  
plt.xticks(np.arange(min(year), max(year)+1, 1.0))  
plt.annotate(round(min(mean\_return),2),  
 (max(year)-.5, min(mean\_return)+.5),fontsize=16,color="red")  
plt.show()

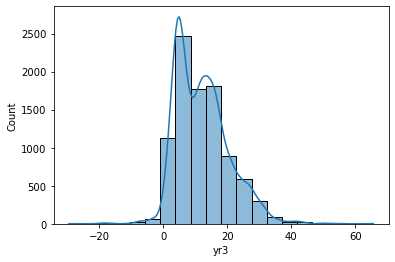


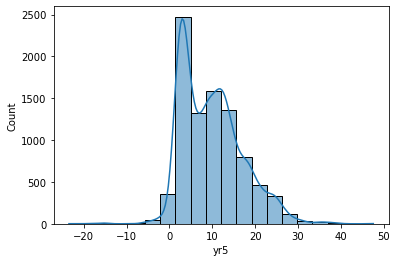
In the above plot, we do see how the average return trends down over time, though it might be asymptotical, and still stay at a positive and acceptable level.

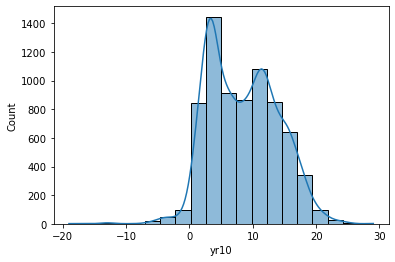
Let us now further explore the distribution of values:

for column in fidelity\_df.columns[3:7]:  
 sns.histplot(fidelity\_df[column], kde=True, bins=20)  
 plt.show()







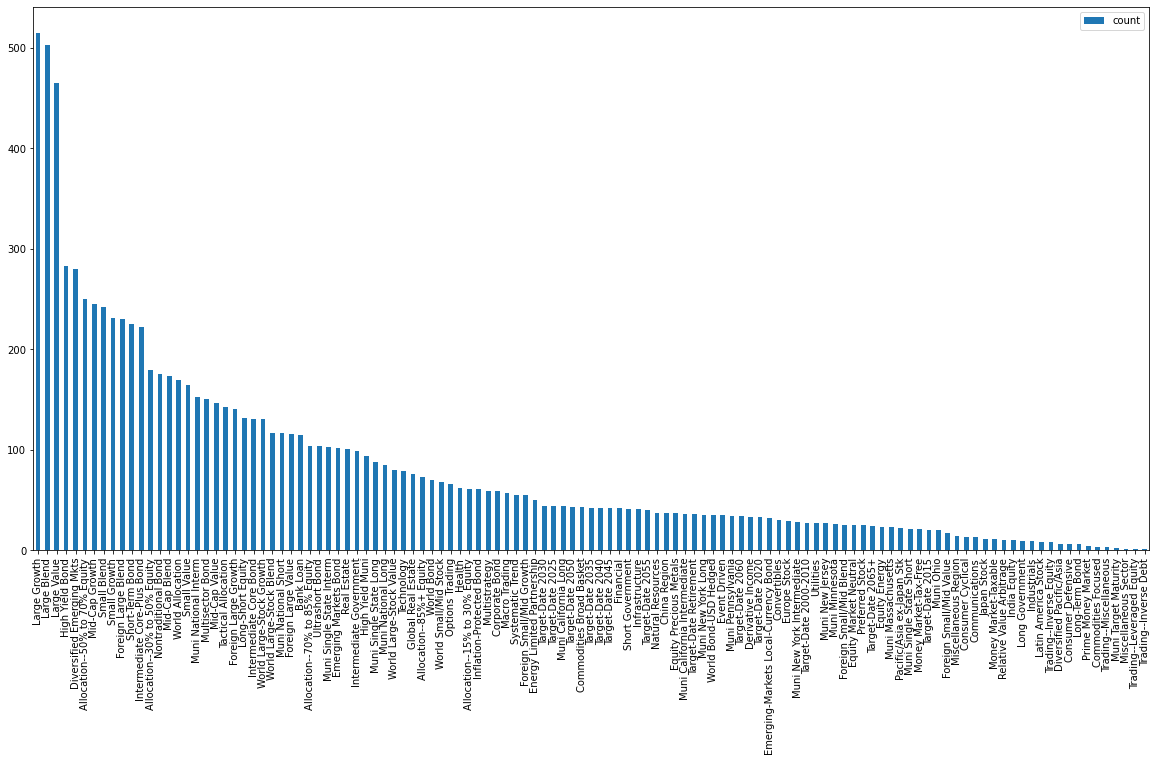


Looking at the distribution of returns, we see most of the mass towards the center, with some skew in some of the horizons. We can also see that the left and right edges of the histograms contain some minimal frequencies. These graphs may approximate normal curves, but not perfectly, as there is visible skew and lowered center values in some cases.

Let's now take a look at the distribution of funds over the Morningstar categories to better understand the market of mutual funds.

from collections import Counter  
  
# Unique categories  
fidelity\_df['morningstar\_category'].nunique()  
  
cat\_cnt = Counter(fidelity\_df['morningstar\_category'])  
cat\_cnt\_fidelity\_df=pd.DataFrame.from\_dict(cat\_cnt, orient='index', columns = ['count'])  
cat\_cnt\_fidelity\_df.sort\_values('count', ascending= False, inplace=True)  
cat\_cnt\_fidelity\_df.plot(kind='bar', figsize = (20,10))

<AxesSubplot:>



len(pd.unique(fidelity\_df['morningstar\_category']))

119

Exploring the above, we can see that there's a main market focus on the 'Large Growth', 'Large Blend', and 'Large Value' with a long tail of the other 116 categories.

# Part 3: Hypothesis Testing

We see that there's a large enough number to perform statistical hypothesis testing. We'll, therefore, perform 4 tests at once, all having the same null and alternative hypothesis, but over the different investment horizons:

* H0: average return <= 2%
* Ha: average return > 2%

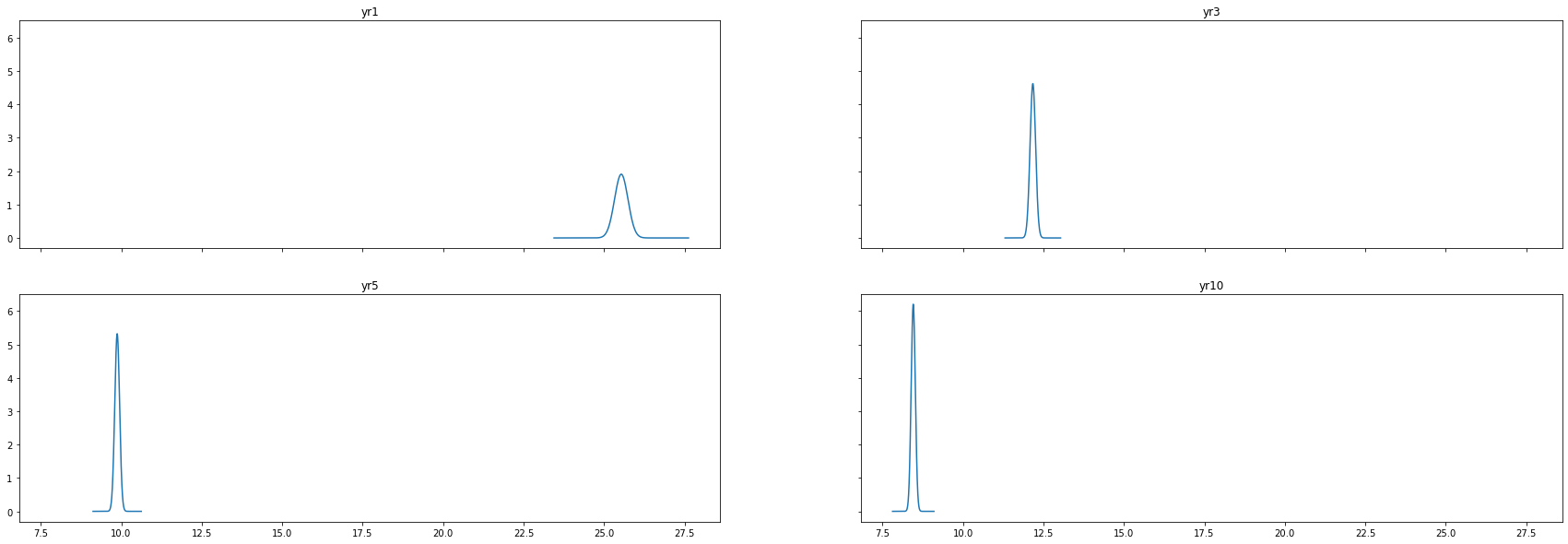
For the 1-year, 3-year, 5-year, and 10-year horizon.

To start, let's explore how the "meta" average behaves considering the data's stdev as a standard error by dividing by the √n of the size of the samples. In other words, how does the measured mean behave or distribute considering the standard error of the data:

from math import \*  
import scipy.stats as st  
  
stats\_df = fidelity\_df.describe().drop(['ytdDaily', 'life\_of\_fund', \  
 'net\_expense\_ratio', 'gross\_expense\_ratio', \  
 'morningstar\_rating\_overall', 'risk', 'std\_dev'\  
 ,'sharpe\_ratio\_3\_yr', 'beta', 'r2', 'minimum\_investment'\  
 , 'last\_dividend'], axis=1)  
stats\_df = stats\_df.drop(['min', '25%', '50%', '75%', 'max'])  
stats\_df

yr1 yr3 yr5 yr10  
count 9520.000000 9204.000000 8922.000000 7283.000000  
mean 25.532315 12.170789 9.872851 8.455235  
std 20.381654 8.276280 7.080595 5.485703

fig, ax = plt.subplots(2,2, sharex=True, sharey=True, figsize=(30,10))  
  
for idx, col in enumerate(stats\_df.columns):  
 count = stats\_df.loc['count'][col]  
 mean = stats\_df.loc['mean'][col]  
 std = stats\_df.loc['std'][col]  
 std\_err = std/sqrt(count)  
   
 x = np.linspace(mean - 10\*std\_err, mean + 10\*std\_err, 1000)  
 ax.flatten()[idx].plot(x, st.norm.pdf(x, mean, std\_err))  
 ax.flatten()[idx].title.set\_text(col)



Now we're able to see the distribution of the average expected return itself for each horizon. Seeing how they all remain absolutely distant from the 2% mark, we're getting a pretty good indication that the Ha or alternative hypothesis from the next part will clearly be accepted.

Not knowing the standard deviation in the wild, we'll perform a T-test on the data.

We randomly stratify the data into 5 groups, and perform 5x4=20 T tests on the data.

stratified\_df = {}  
for i in range(5):  
 stratified\_df[i] = fidelity\_df[fidelity\_df.columns[3:7]].sample(frac=0.2)  
 print(f'Random sample test #{i+1}')  
 print(stratified\_df[i].apply(  
 lambda period\_sample:   
 st.ttest\_1samp(period\_sample.dropna(), 2, alternative='greater' )) \  
 .rename({0:'T Statistic', 1:'P Value'}))  
 print()

Random sample test #1  
 yr1 yr3 yr5 yr10  
T Statistic 51.240158 53.389997 4.736050e+01 4.559708e+01  
P Value 0.000000 0.000000 2.081228e-317 1.368278e-282  
  
Random sample test #2  
 yr1 yr3 yr5 yr10  
T Statistic 49.568869 51.996934 4.696277e+01 4.627493e+01  
P Value 0.000000 0.000000 1.000833e-314 7.718880e-289  
  
Random sample test #3  
 yr1 yr3 yr5 yr10  
T Statistic 51.469627 53.424833 4.771945e+01 4.566700e+01  
P Value 0.000000 0.000000 1.225283e-321 9.157987e-283  
  
Random sample test #4  
 yr1 yr3 yr5 yr10  
T Statistic 50.668846 53.173947 4.748732e+01 4.566009e+01  
P Value 0.000000 0.000000 5.072078e-320 2.829554e-284  
  
Random sample test #5  
 yr1 yr3 yr5 yr10  
T Statistic 49.173508 51.399747 4.615293e+01 4.387022e+01  
P Value 0.000000 0.000000 4.162747e-307 1.959979e-268

# Conclusion

Seeing these minimal P-Values for our one-sample, one-sided T-Tests, we're able to reject the null hypothesis on all time periods, thus safely assuming that mutual funds will have an average return of more than 2% on any of the 1-year, 3-year, 5-year, and 10-year investment horizons.

# Part 4 Bonus: Regression Analysis / ANOVA Analysis (extra 20 points)

We will submit this part in a separate document.

# Function for forward select linear model

import statsmodels.formula.api as smf  
  
def forward\_selected(data, response):  
 """Linear model designed by forward selection.  
  
 Parameters:  
 -----------  
 data : pandas DataFrame with all possible predictors and response  
  
 response: string, name of response column in data  
  
 Returns:  
 --------  
 model: an "optimal" fitted statsmodels linear model  
 with an intercept  
 selected by forward selection  
 evaluated by adjusted R-squared  
 """  
 remaining = set(data.columns)  
 remaining.remove(response)  
 selected = []  
 current\_score, best\_new\_score = 0.0, 0.0  
 while remaining and current\_score == best\_new\_score:  
 scores\_with\_candidates = []  
 for candidate in remaining:  
 formula = "{} ~ {}".format(response,  
 ' + '.join(selected + [candidate]))  
 score = smf.ols(formula, data).fit().rsquared\_adj  
 scores\_with\_candidates.append((score, candidate))  
 scores\_with\_candidates.sort()  
 best\_new\_score, best\_candidate = scores\_with\_candidates.pop()  
 if current\_score < best\_new\_score:  
 remaining.remove(best\_candidate)  
 selected.append(best\_candidate)  
 current\_score = best\_new\_score  
 formula = "{} ~ {}".format(response,  
 ' + '.join(selected))  
 model = smf.ols(formula, data).fit()  
 return model

# Importing the csv  
df= pd.read\_csv('fidelity\_mutual\_funds\_return\_w\_risk.csv')  
df.head()

name morningstar\_category \  
0 Baron Partners Fund Institutional Shares (BPTIX) Large Growth   
1 Baron Partners Fund Retail Shares (BPTRX) Large Growth   
2 Morgan Stanley Institutional Fund, Inc. Incept... Small Growth   
3 Morgan Stanley Institutional Fund, Inc. Incept... Small Growth   
4 Morgan Stanley Institutional Fund, Inc. Incept... Small Growth   
  
 ytdDaily yr1 yr3 yr5 yr10 life\_of\_fund net\_expense\_ratio \  
0 44.60 110.27 65.56 47.54 29.02 27.02 1.30   
1 44.28 109.72 65.13 47.15 28.68 20.45 1.56   
2 20.05 81.15 55.11 38.37 22.37 14.14 1.00   
3 19.75 80.67 54.70 37.98 22.01 13.83 1.35   
4 NaN 79.39 53.46 36.90 21.11 13.00 2.10   
  
 gross\_expense\_ratio morningstar\_rating\_overall risk std\_dev \  
0 1.30 1137.0 6 40.41   
1 1.56 1137.0 6 40.39   
2 1.19 574.0 7 40.44   
3 1.45 574.0 7 40.48   
4 2.27 574.0 7 40.42   
  
 sharpe\_ratio\_3\_yr beta r2 minimum\_investment last\_dividend   
0 1.60 1.51 0.63 1000000.0 0.2224   
1 1.59 1.51 0.63 2500.0 0.1243   
2 1.34 1.40 0.71 5000000.0 0.0000   
3 1.32 1.40 0.71 2500.0 0.0000   
4 1.30 1.40 0.71 2500.0 NaN

df.columns

Index(['name', 'morningstar\_category', 'ytdDaily', 'yr1', 'yr3', 'yr5', 'yr10',  
 'life\_of\_fund', 'net\_expense\_ratio', 'gross\_expense\_ratio',  
 'morningstar\_rating\_overall', 'risk', 'std\_dev', 'sharpe\_ratio\_3\_yr',  
 'beta', 'r2', 'minimum\_investment', 'last\_dividend'],  
 dtype='object')

# Feature selection

# Extracting all the numeric columns  
df\_num = df[['ytdDaily', 'yr1', 'yr3', 'yr5', 'yr10',  
 'life\_of\_fund', 'net\_expense\_ratio', 'gross\_expense\_ratio',  
 'morningstar\_rating\_overall', 'risk', 'std\_dev', 'sharpe\_ratio\_3\_yr',  
 'beta', 'r2', 'minimum\_investment', 'last\_dividend']].dropna()  
df\_num

ytdDaily yr1 yr3 yr5 yr10 life\_of\_fund net\_expense\_ratio \  
0 44.60 110.27 65.56 47.54 29.02 27.02 1.30   
1 44.28 109.72 65.13 47.15 28.68 20.45 1.56   
2 20.05 81.15 55.11 38.37 22.37 14.14 1.00   
3 19.75 80.67 54.70 37.98 22.01 13.83 1.35   
5 29.85 78.53 51.07 37.87 21.75 21.12 1.07   
... ... ... ... ... ... ... ...   
9190 50.49 112.78 -11.17 -9.38 -4.58 -1.65 1.35   
9192 37.18 106.34 -16.52 -14.67 -12.41 -3.39 1.42   
9193 36.87 105.81 -16.69 -14.87 -12.72 -3.74 1.68   
9194 36.89 105.82 -16.74 -14.88 -12.62 -3.63 1.68   
9195 36.04 104.30 -17.35 -15.51 -13.28 -4.34 2.43   
  
 gross\_expense\_ratio morningstar\_rating\_overall risk std\_dev \  
0 1.30 1137.0 6 40.41   
1 1.56 1137.0 6 40.39   
2 1.19 574.0 7 40.44   
3 1.45 574.0 7 40.48   
5 1.07 550.0 6 33.96   
... ... ... ... ...   
9190 1.87 72.0 8 48.41   
9192 1.42 72.0 8 56.46   
9193 1.68 72.0 8 56.42   
9194 1.68 72.0 8 56.44   
9195 2.43 72.0 8 56.40   
  
 sharpe\_ratio\_3\_yr beta r2 minimum\_investment last\_dividend   
0 1.60 1.51 0.63 1000000.0 0.222400   
1 1.59 1.51 0.63 2500.0 0.124300   
2 1.34 1.40 0.71 5000000.0 0.000000   
3 1.32 1.40 0.71 2500.0 0.000000   
5 1.47 1.16 0.62 1000000.0 0.002900   
... ... ... ... ... ...   
9190 -0.25 1.12 0.95 2500.0 0.112000   
9192 -0.31 2.44 0.64 2500.0 1.837037   
9193 -0.31 2.44 0.64 2500.0 1.837037   
9194 -0.32 2.44 0.64 2500.0 1.837037   
9195 -0.33 2.44 0.64 2500.0 1.837037   
  
[6203 rows x 16 columns]

# Linear regression model

model = forward\_selected(df\_num.dropna(), 'yr10')  
print("Selected features for the model:")  
print(model.model.formula)  
print("----------------------------------")  
print("Adjusted R squared for the model:")  
print(model.rsquared\_adj)

Selected features for the model:  
yr10 ~ yr5 + yr3 + life\_of\_fund + ytdDaily + yr1 + r2 + gross\_expense\_ratio + morningstar\_rating\_overall + risk + sharpe\_ratio\_3\_yr + std\_dev + net\_expense\_ratio + minimum\_investment  
----------------------------------  
Adjusted R squared for the model:  
0.9444266011578509

print(model.summary())

OLS Regression Results   
==============================================================================  
Dep. Variable: yr10 R-squared: 0.945  
Model: OLS Adj. R-squared: 0.944  
Method: Least Squares F-statistic: 8109.  
Date: Sat, 20 Nov 2021 Prob (F-statistic): 0.00  
Time: 12:31:00 Log-Likelihood: -10247.  
No. Observations: 6203 AIC: 2.052e+04  
Df Residuals: 6189 BIC: 2.062e+04  
Df Model: 13   
Covariance Type: nonrobust   
==============================================================================================  
 coef std err t P>|t| [0.025 0.975]  
----------------------------------------------------------------------------------------------  
Intercept 0.7433 0.105 7.070 0.000 0.537 0.949  
yr5 0.9030 0.011 83.411 0.000 0.882 0.924  
yr3 -0.2693 0.010 -27.497 0.000 -0.288 -0.250  
life\_of\_fund 0.2965 0.008 36.702 0.000 0.281 0.312  
ytdDaily 0.0836 0.005 15.369 0.000 0.073 0.094  
yr1 -0.0261 0.004 -6.132 0.000 -0.035 -0.018  
r2 0.7745 0.069 11.274 0.000 0.640 0.909  
gross\_expense\_ratio -0.2662 0.033 -8.118 0.000 -0.330 -0.202  
morningstar\_rating\_overall 0.0003 5.25e-05 4.770 0.000 0.000 0.000  
risk -0.0632 0.020 -3.179 0.001 -0.102 -0.024  
sharpe\_ratio\_3\_yr -0.5183 0.079 -6.576 0.000 -0.673 -0.364  
std\_dev -0.0300 0.006 -4.701 0.000 -0.043 -0.017  
net\_expense\_ratio 0.1494 0.048 3.093 0.002 0.055 0.244  
minimum\_investment -3.352e-08 2.01e-08 -1.665 0.096 -7.3e-08 5.95e-09  
==============================================================================  
Omnibus: 1135.566 Durbin-Watson: 1.841  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 8786.598  
Skew: -0.662 Prob(JB): 0.00  
Kurtosis: 8.678 Cond. No. 6.40e+06  
==============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
[2] The condition number is large, 6.4e+06. This might indicate that there are  
strong multicollinearity or other numerical problems.

# Correlation between the response and explanatory variables

for col in df\_num.columns.difference(['yr10', 'last\_dividend', 'beta']):  
 print(f"Correlation between yr10 and {col} is {stats.pearsonr(df\_num['yr10'].values,df\_num[col].values)[0]}")

Correlation between yr10 and gross\_expense\_ratio is -0.043466501612530156  
Correlation between yr10 and life\_of\_fund is 0.8252743569158877  
Correlation between yr10 and minimum\_investment is 0.03846300222625589  
Correlation between yr10 and morningstar\_rating\_overall is 0.5347603693302172  
Correlation between yr10 and net\_expense\_ratio is 0.0037295280795975154  
Correlation between yr10 and r2 is 0.3601893515895218  
Correlation between yr10 and risk is 0.5510176060654153  
Correlation between yr10 and sharpe\_ratio\_3\_yr is 0.42204222818958054  
Correlation between yr10 and std\_dev is 0.5833578563997679  
Correlation between yr10 and yr1 is 0.6612711971272358  
Correlation between yr10 and yr3 is 0.8749152051615144  
Correlation between yr10 and yr5 is 0.9461939495764184  
Correlation between yr10 and ytdDaily is 0.6567130333247719

# Actual vs predicted plot

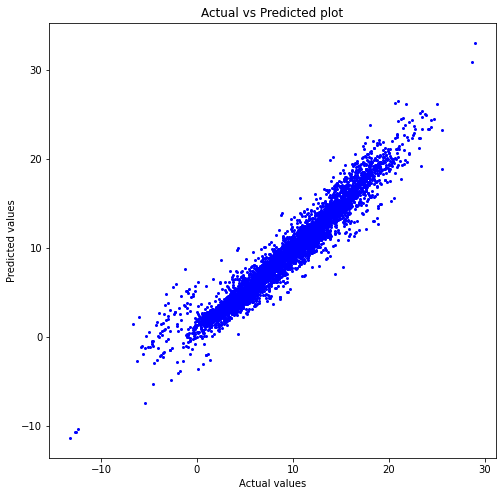
X = df\_num[df\_num.columns.difference(['yr10', 'last\_dividend', 'beta'])].dropna()  
y = df\_num['yr10'].dropna().values  
predictions = model.predict(X).values  
r2 = model.rsquared\_adj

correlation, p\_value = stats.pearsonr(y,predictions)  
print(f"Correlation between actual and predicted is {correlation}")

Correlation between actual and predicted is 0.9718760664044803

plt.figure(figsize=(8,8))  
plt.plot(y, predictions, 'o', color='blue', markersize=2)  
plt.xlabel("Actual values")  
plt.ylabel("Predicted values")  
plt.title('Actual vs Predicted plot')

Text(0.5, 1.0, 'Actual vs Predicted plot')



#### Our model's predictions are pretty close to actual values. Also our R-squared value is 94.5% which is pretty high.

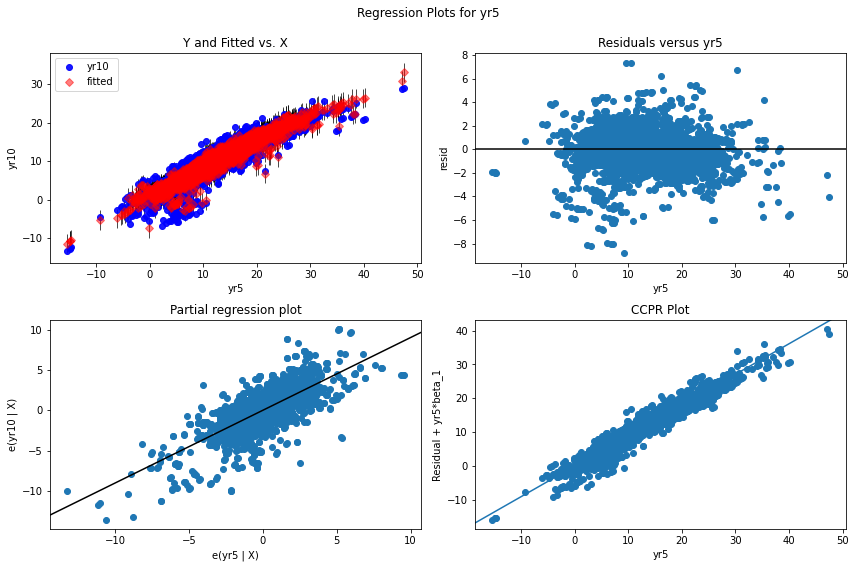
## Model score

from sklearn.linear\_model import LinearRegression  
lin\_reg\_model = LinearRegression()  
  
lin\_reg\_model.fit(X, y)  
print(f"Training Data Score: {lin\_reg\_model.score(X, y)}")

Training Data Score: 0.9445430884498451

# Model plot with one explanatory variable with the highest correlation

#define figure size  
fig = plt.figure(figsize=(12,8))  
  
#produce regression plots  
fig = sm.graphics.plot\_regress\_exog(model, 'yr5', fig=fig)



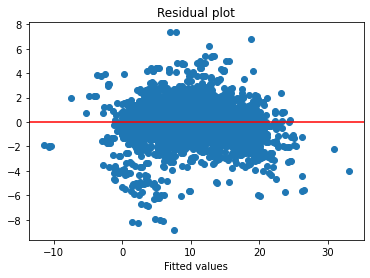
## Residual plot

# Getting residual values  
residuals = y - predictions  
residuals

array([-4.01766782, -2.1700478 , 0.06551326, ..., -2.0163843 ,  
 -1.95942283, -1.89180646])

plt.scatter(predictions, residuals)  
plt.axhline(y = 0, color = 'r', linestyle = '-')  
plt.xlabel("Fitted values")  
plt.title('Residual plot')

Text(0.5, 1.0, 'Residual plot')

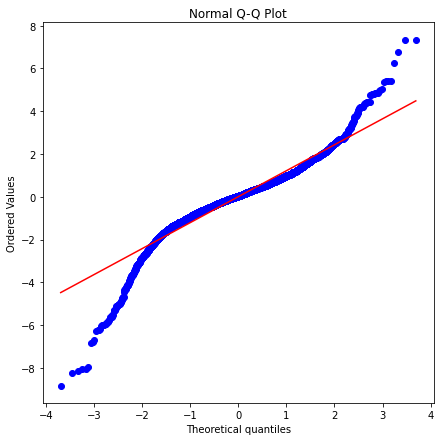


#### From the above plot we observe that the residual against yr10 is slightly curved as most of the data points are below horizontal line. This indicates that a non-linear relation might have given us a better model for this dataset.

## Normal Q-Q Plot

# Plotting residual values on a normall Q-Q plot  
plt.figure(figsize=(7,7))  
stats.probplot(residuals, dist="norm", plot=plt)  
plt.title("Normal Q-Q Plot")

Text(0.5, 1.0, 'Normal Q-Q Plot')



#### From the above plot, we can say that the distribution of residuals is pretty normal.

# Anova of OLS (Ordinary least squared) model

anova\_table = sm.stats.anova\_lm(model)  
anova\_table

df sum\_sq mean\_sq \  
yr5 1.0 159572.166398 159572.166398   
yr3 1.0 4072.959165 4072.959165   
life\_of\_fund 1.0 3199.929739 3199.929739   
ytdDaily 1.0 669.963411 669.963411   
yr1 1.0 365.704202 365.704202   
r2 1.0 188.684904 188.684904   
gross\_expense\_ratio 1.0 111.082308 111.082308   
morningstar\_rating\_overall 1.0 38.323975 38.323975   
risk 1.0 27.740343 27.740343   
sharpe\_ratio\_3\_yr 1.0 50.175736 50.175736   
std\_dev 1.0 32.363664 32.363664   
net\_expense\_ratio 1.0 18.596081 18.596081   
minimum\_investment 1.0 4.427257 4.427257   
Residual 6189.0 9884.449514 1.597100   
  
 F PR(>F)   
yr5 99913.721691 0.000000e+00   
yr3 2550.222371 0.000000e+00   
life\_of\_fund 2003.588073 0.000000e+00   
ytdDaily 419.487554 2.934155e-90   
yr1 228.980208 8.004859e-51   
r2 118.142226 2.842844e-27   
gross\_expense\_ratio 69.552523 9.083572e-17   
morningstar\_rating\_overall 23.995983 9.899106e-07   
risk 17.369200 3.119681e-05   
sharpe\_ratio\_3\_yr 31.416786 2.171539e-08   
std\_dev 20.264023 6.869318e-06   
net\_expense\_ratio 11.643657 6.483453e-04   
minimum\_investment 2.772061 9.597322e-02   
Residual NaN NaN

#### F critical value

# At 5% level of significance   
stats.f.ppf(q=1-.05, dfn=13, dfd=6189)

1.7217356615435946

#### So, from the anova table above, we can conclude that all the explanatory variables have significant variation as their F statistical values are larger than F critical value and they are all statistically significant.

# Anova for different years' rates

stats.f\_oneway(df\_num['yr1'], df\_num['yr3'], df\_num['yr5'], df\_num['yr10'])

F\_onewayResult(statistic=2917.3689315577753, pvalue=0.0)

#### As the p-value is close to 0, we can say that the mean returns of each years are different