Tradexa_Assignment

December 29, 2025

[10]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression



print("Libraries Loaded")
```

```
Libraries Loaded
```

[2]:
```
df = pd.read_csv("fund_data.csv")
df.head()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[2], line 1
----> 1 df = pd.read_csv(                )
      2 df.head()

File ~/jupyter-env/lib/python3.12/site-packages/pandas/io/parsers/readers.py:
↪1026, in read_csv(filepath_or_buffer, sep, delimiter, header, names,␣
↪index_col, usecols, dtype, engine, converters, true_values, false_values,␣
↪skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na,␣
↪na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format,␣
↪keep_date_col, date_parser, date_format, dayfirst, cache_dates, iterator,␣
↪chunksize, compression, thousands, decimal, lineterminator, quotechar,␣
↪quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect␣
↪on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision,␣
↪storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (…)   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File ~/jupyter-env/lib/python3.12/site-packages/pandas/io/parsers/readers.py:
↪620, in _read(filepath_or_buffer, kwds)
```

```
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File ~/jupyter-env/lib/python3.12/site-packages/pandas/io/parsers/readers.py:
 ↪1620, in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File ~/jupyter-env/lib/python3.12/site-packages/pandas/io/parsers/readers.py:
 ↪1880, in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get(           , None),
   1884     compression=self.options.get(              , None),
   1885     memory_map=self.options.get(             , False),
   1886     is_text=is_text,
   1887     errors=self.options.get(                 ,          ),
   1888     storage_options=self.options.get(                  , None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File ~/jupyter-env/lib/python3.12/site-packages/pandas/io/common.py:873, in␣
 ↪get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text,␣
 ↪errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline=  ,
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'fund_data.csv'
```

[3]:
```
import os
os.getcwd()
```

[3]:
```
'/home/suyash'
```

[4]:
```
df = pd.read_csv("fund_data.csv")
df.head()
```

[4]:
```
                FundName MarketCap    Type      Risk  SharpeRatio  1YrReturn%  \
0     Alpha Equity Fund     Large  Equity      High          1.2        18.5
1  Bluechip Growth Fund     Large  Equity  Moderate          1.0        15.3
2  Midcap Opportunities       Mid  Equity      High          1.4        22.1
3    Balanced Advantage     Large  Hybrid  Moderate          0.9        12.4
4    Smallcap Discovery     Small  Equity      High          1.6        28.3

   3YrReturn%
0        14.2
1        12.8
2        17.6
3        10.1
4        21.4
```

[5]:
```
# Drop rows with missing critical values
df = df.dropna(subset=['MarketCap', 'Type', 'Risk', '3YrReturn%',␣
 ↪'SharpeRatio', '1YrReturn%'])

# Convert percentage columns to numeric
df['3YrReturn%'] = pd.to_numeric(df['3YrReturn%'], errors='coerce')
df['1YrReturn%'] = pd.to_numeric(df['1YrReturn%'], errors='coerce')
```

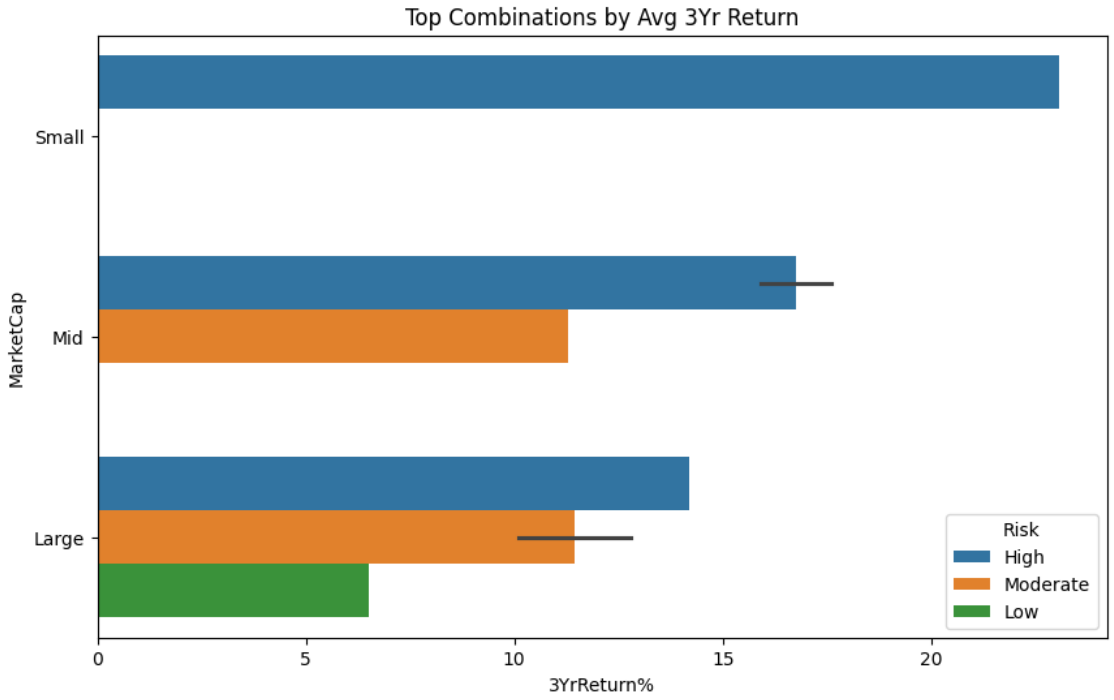
[6]:
```
grouped = (
    df.groupby(['MarketCap', 'Type', 'Risk'])['3YrReturn%']
    .mean()
    .reset_index()
)

# Find best combination
best_combo = grouped.loc[grouped['3YrReturn%'].idxmax()]
print("Best Combination for 3Yr Return:\n", best_combo)
```

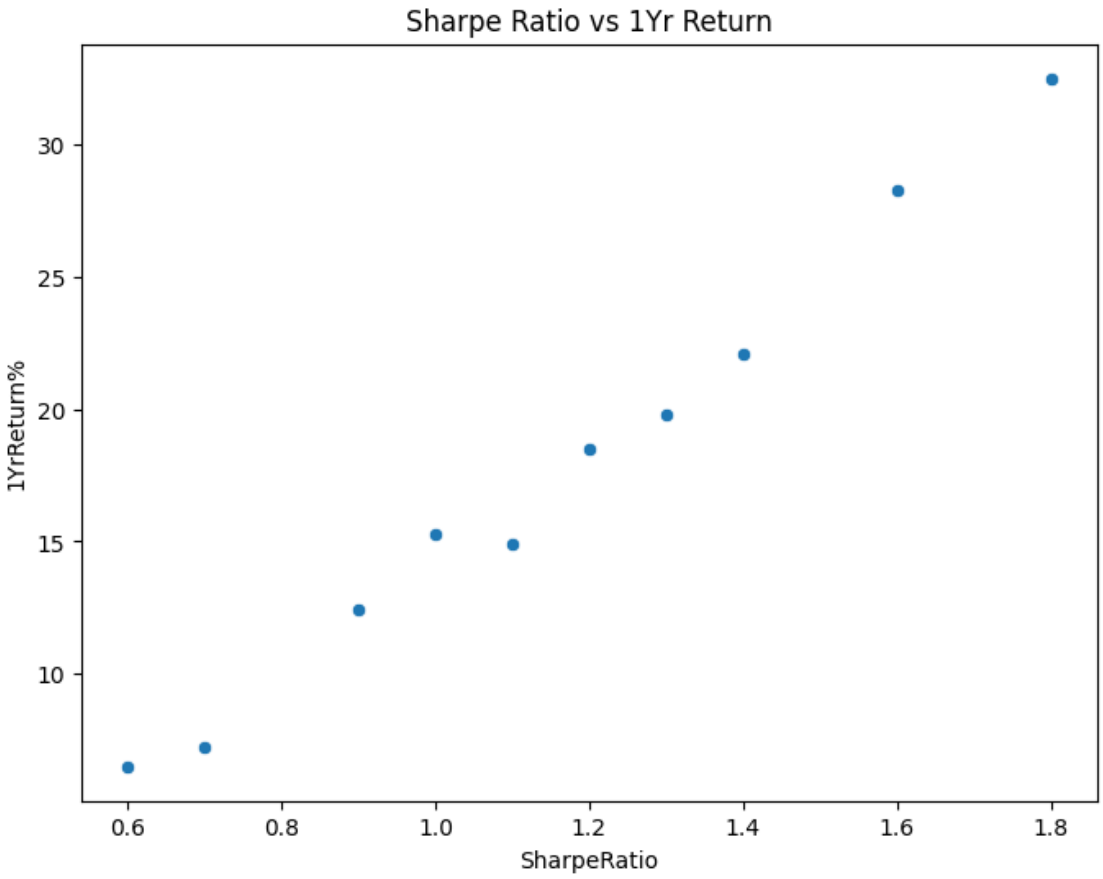
```
Best Combination for 3Yr Return:
 MarketCap      Small
Type           Equity
Risk             High
3YrReturn%      23.05
```

```
Name: 7, dtype: object
```

[7]:
```
plt.figure(figsize=(10,6))
sns.barplot(
    data=grouped.sort_values('3YrReturn%', ascending=False).head(10),
    x='3YrReturn%',
    y='MarketCap',
    hue='Risk'
)
plt.title("Top Combinations by Avg 3Yr Return")
plt.show()
```



[8]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x='SharpeRatio', y='1YrReturn%')
plt.title("Sharpe Ratio vs 1Yr Return")
plt.show()
```



[11]:
```
X = df[['SharpeRatio']]
y = df['1YrReturn%']

# Polynomial regression (degree 2)
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

model = LinearRegression()
model.fit(X_poly, y)
```

[11]:
```
LinearRegression()
```

[ ]:
```

```