SUMMARY OF ASSIGNMENT

-KEY CONTENTS

1. Tools and libraries used
2. Models & Techniques implemented:
3. Visualization Techniques used:

-SUMMARY-EACH STEP WITH CORRESPONDING CODE

1. objective
2. Data pre-processing
3. Data Cleaning
4. Problem 1: Optimal 3-Year Return Combination(methodology, visualization, outcome)
5. Problem 2: Optimal Sharpe Ratio for 1-Year Return(methodology, Finding optimal sharpe ratio,visualization,outcome)

-ENTIRE CODE

Tools & Libraries

Python Libraries used:

-pandas, numpy – Data handling and numerical computations.
-matplotlib, seaborn – Data visualization.
-scikit-learn – Machine learning, preprocessing, and evaluation.

Models & Techniques implemented:

-GroupBy and aggregation for summary statistics.
-Random Forest Regressor for predicting 1-Year Return.
-Feature preprocessing: StandardScaler and OneHotEncoder.
-Grid search approach for optimizing Sharpe Ratio.

Visualization Techniques used:

-Horizontal bar charts for categorical comparison.
-Line plots to demonstrate predicted returns vs Sharpe Ratio.

Key Insights

-Top 3-Year Return combination highlights which Market Cap, Fund Type, and Risk profile consistently performs well.
-Optimal Sharpe Ratio provides a numeric benchmark for maximizing 1-Year Return while considering risk-adjusted performance.
-This analysis can guide investment decisions, fund selection, and portfolio optimization.

Assignment – Summary-each step is explained with its corresponding code.

Objective:

-The goal of this assignment was to analyze an investment dataset and provide

actionable insights, focusing on:
-Identifying the ideal combination of Market Cap, Fund Type, and Risk that yields the highest 3-Year Return (%).
-Determining the optimal Sharpe Ratio to maximize 1-Year Return (%).

Data Preprocessing:

-Importing the necessary libraries
-Dataset: CSV file (Dataset.xlsx) containing features like Market Cap, Fund Type, Risk, 1-Year Return, 3-Year Return, and Sharpe Ratio.

```
# IMPORT LIBRARIES
# -------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_absolute_error

# -------------------------------
# STEP 1: LOAD DATASET
# -------------------------------
# Read CSV
df = pd.read_excel("Dataset.xlsx")
```

Cleaning Steps:

-Removed duplicates and rows with missing essential values.
-Standardized column names (removed spaces, special characters).
-Converted percentage columns (e.g., 3YrReturn%, 1YrReturn%) from strings to numeric.
-Standardized text columns (MarketCap, Type, Risk) with consistent capitalization.

```
# Clean column names (remove spaces, special chars, standardize)
df.columns = df.columns.str.strip().str.replace(" ", "").str.replace("-",
"").str.replace("_", "")
print("Cleaned Column Names:", df.columns.tolist())

# Preview first rows
print("\nSample Data:")
print(df.head())

# -------------------------------
# STEP 2: DATA CLEANING
# -------------------------------
# Convert % columns to numeric (if stored as strings like "12.5%")
percent_cols = [c for c in df.columns if 'Return' in c or 'Sharpe' in c]
for col in percent_cols:
    df[col] = df[col].astype(str).str.replace('%', '').str.replace(',', '')
```

```
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Identify key columns (adjust if your dataset uses different names)
market_col = 'MarketCap'
type_col = 'Type'
risk_col = 'Risk'
return_3yr_col = [c for c in df.columns if '3YrReturn' in c][0]
return_1yr_col = [c for c in df.columns if '1YrReturn' in c][0]
sharpe_col = [c for c in df.columns if 'Sharpe' in c][0]

# Drop rows with missing essential values
df = df.dropna(subset=[market_col, type_col, risk_col, return_3yr_col,
return_1yr_col, sharpe_col])

# Standardize text columns
for col in [market_col, type_col, risk_col]:
    df[col] = df[col].str.title().str.strip()
```

Problem 1: Optimal 3-Year Return

Combination Methodology:

-Grouped data by MarketCap, Type, and Risk.
-Calculated the mean 3-Year Return for each combination.
-Sorted combinations to find the top-performing one.

Visualization:

Created a horizontal bar chart of the top 10 combinations for 3-Year Return.

Outcome:

The best MarketCap|Type|Risk combination was identified.
Provides insights into which fund types and risk levels maximize long-term returns.

```
# STEP 3: PROBLEM 1 — BEST COMBINATION FOR 3-YR RETURN
# -------------------------------
# Group by MarketCap, Type, Risk and calculate mean 3YrReturn
grouped = df.groupby([market_col, type_col, risk_col])
[return_3yr_col].mean().reset_index()
grouped = grouped.sort_values(by=return_3yr_col, ascending=False)

best_combo = grouped.iloc[0]
print("\n  Ideal Combination for Highest 3-Year Return:")
print(best_combo)

# Visualization: Top 10 combinations
top10 = grouped.head(10)
plt.figure(figsize=(10,6))
sns.barplot(
    y=top10.apply(lambda x: f"{x[market_col]} | {x[type_col]} | {x[risk_col]}",
axis=1),
    x=top10[return_3yr_col],
    palette="Blues_r"
)
```

```
plt.xlabel("Average 3-Year Return (%)")
plt.ylabel("MarketCap | Type | Risk")
plt.title("Top 10 Combinations by 3-Year Return")
plt.tight_layout()
plt.savefig("top_3yr_combos.png", dpi=150)
plt.close()
```

Problem 2: Optimal Sharpe Ratio for 1-Year

Return Methodology:

-Built a Random Forest Regressor to predict 1-Year Return using:
 Features: Sharpe Ratio (numeric), Market Cap, Type, and Risk (categorical). Preprocessing: StandardScaler for numeric features, OneHotEncoder for categorical features.

Trained/test split: 80/20.

Predicted 1-Year Return for unseen test data to evaluate model performance.

Finding Optimal Sharpe Ratio:

-Created a baseline profile with median/mode values for other features.
-Varied the Sharpe Ratio across a grid and predicted 1-Year Return.
-Selected the Sharpe Ratio that maximized predicted 1-Year Return.

Visualization:

Line plot of Predicted 1-Year Return vs Sharpe Ratio, with optimal Sharpe highlighted.

Outcome:

Provided an actionable Sharpe Ratio recommendation for investors targeting maximum short-term return.

```
STEP 4: PROBLEM 2 — OPTIMAL SHARPE RATIO FOR 1-YR RETURN
# -------------------------------
# Features: include Sharpe Ratio and categorical variables
X = df[[sharpe_col, market_col, type_col, risk_col]]
y = df[return_1yr_col]

# Preprocessing pipelines
numeric_features = [sharpe_col]
categorical_features = [market_col, type_col, risk_col]

numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numeric_features),
```

```
    ('cat', categorical_transformer, categorical_features)
])

# Random Forest Regressor pipeline
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(random_state=42, n_estimators=200))
])

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)
print(f"\n1-Year Return Prediction - Test R2: {r2_score(y_test, y_pred):.4f},
MAE: {mean_absolute_error(y_test, y_pred):.4f}")

# -------------------------------
# Find optimal Sharpe Ratio
# -------------------------------
# Create baseline (median numeric, mode categorical)
baseline = {}
baseline[sharpe_col] = 0  # placeholder, will vary
for col in categorical_features:
    baseline[col] = X[col].mode()[0]

# Generate a Sharpe ratio grid
sharpe_min, sharpe_max = X[sharpe_col].min(), X[sharpe_col].max()
sharpe_grid = np.linspace(sharpe_min, sharpe_max, 200)
pred_returns = []

for s in sharpe_grid:
    row = baseline.copy()
    row[sharpe_col] = s
    row_df = pd.DataFrame([row])
    pred = model.predict(row_df)[0]
    pred_returns.append(pred)

pred_returns = np.array(pred_returns)
best_idx = pred_returns.argmax()
optimal_sharpe = sharpe_grid[best_idx]
optimal_return = pred_returns[best_idx]

# Visualization
plt.figure(figsize=(10,6))
plt.plot(sharpe_grid, pred_returns, label='Predicted 1YrReturn')
plt.axvline(optimal_sharpe, color='r', linestyle='--', label=f'Optimal Sharpe ≈
{optimal_sharpe:.3f}')
plt.xlabel("Sharpe Ratio")
plt.ylabel("Predicted 1-Year Return (%)")
plt.title("Predicted 1-Year Return vs Sharpe Ratio")
plt.legend()
plt.tight_layout()
plt.savefig("sharpe_vs_1yrreturn.png", dpi=150)
plt.close()

print(f"\n  Optimal Sharpe Ratio to maximize 1-Year Return:
{optimal_sharpe:.4f}")
print(f"Predicted 1-Year Return at optimal Sharpe: {optimal_return:.4f}%")

# -------------------------------
# ANALYSIS SUMMARY
```

```
# -------------------------------
print("\n  Analysis Summary:")
print(f"- Best MarketCap|Type|Risk combination (3YrReturn):
{best_combo[market_col]} | {best_combo[type_col]} | {best_combo[risk_col]}")
print(f"- Predicted 3YrReturn: {best_combo[return_3yr_col]:.2f}%")
print(f"- Optimal Sharpe Ratio to maximize 1YrReturn: {optimal_sharpe:.4f}")
print(f"- Predicted 1YrReturn at optimal Sharpe: {optimal_return:.2f}%")
```

ENTIRE CODE

```
"""
For this Assignemnet I have used Jupyter notebook

lets call it Data_analysis_python_code.py/Data_analysis_python_code.ipynb

Description:
    - This script analyzes an investment dataset to:
        1. Find the ideal combination of Market Cap, Type, and Risk for maximum
3-Year Return (%)
        2. Determine the optimal Sharpe Ratio to maximize 1-Year Return (%)
    - Includes visualizations and machine learning modeling.

Usage:
    - Placing the dataset (XLSX) in the same folder and name it "dataset.xlsx"
    - Run: python investment_analysis.py
Outputs:
    - Prints analysis summary to console
    - Saves plots:
        - 'top_3yr_combos.png'
        - 'sharpe_vs_1yrreturn.png'
"""




# -------------------------------
# IMPORT LIBRARIES
# -------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_absolute_error

# -------------------------------
# STEP 1: LOAD DATASET
# -------------------------------
# Read CSV
df = pd.read_excel("Dataset.xlsx")

# Clean column names (remove spaces, special chars, standardize)
df.columns = df.columns.str.strip().str.replace(" ", "").str.replace("-",
"").str.replace("_", "")
print("Cleaned Column Names:", df.columns.tolist())

# Preview first rows
```

```
print("\nSample Data:")
print(df.head())

# --------------------------------
# STEP 2: DATA CLEANING
# --------------------------------
# Convert % columns to numeric (if stored as strings like "12.5%")
percent_cols = [c for c in df.columns if 'Return' in c or 'Sharpe' in c]
for col in percent_cols:
    df[col] = df[col].astype(str).str.replace('%', '').str.replace(',', '')
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Identify key columns (adjust if your dataset uses different names)
market_col = 'MarketCap'
type_col = 'Type'
risk_col = 'Risk'
return_3yr_col = [c for c in df.columns if '3YrReturn' in c][0]
return_1yr_col = [c for c in df.columns if '1YrReturn' in c][0]
sharpe_col = [c for c in df.columns if 'Sharpe' in c][0]

# Drop rows with missing essential values
df = df.dropna(subset=[market_col, type_col, risk_col, return_3yr_col,
return_1yr_col, sharpe_col])

# Standardize text columns
for col in [market_col, type_col, risk_col]:
    df[col] = df[col].str.title().str.strip()

# --------------------------------
# STEP 3: PROBLEM 1 — BEST COMBINATION FOR 3-YR RETURN
# --------------------------------
# Group by MarketCap, Type, Risk and calculate mean 3YrReturn
grouped = df.groupby([market_col, type_col, risk_col])
[return_3yr_col].mean().reset_index()
grouped = grouped.sort_values(by=return_3yr_col, ascending=False)

best_combo = grouped.iloc[0]
print("\n  Ideal Combination for Highest 3-Year Return:")
print(best_combo)

# Visualization: Top 10 combinations
top10 = grouped.head(10)
plt.figure(figsize=(10,6))
sns.barplot(
    y=top10.apply(lambda x: f"{x[market_col]} | {x[type_col]} | {x[risk_col]}",
axis=1),
    x=top10[return_3yr_col],
    palette="Blues_r"
)
plt.xlabel("Average 3-Year Return (%)")
plt.ylabel("MarketCap | Type | Risk")
plt.title("Top 10 Combinations by 3-Year Return")
plt.tight_layout()
plt.savefig("top_3yr_combos.png", dpi=150)
plt.close()

# --------------------------------
# STEP 4: PROBLEM 2 — OPTIMAL SHARPE RATIO FOR 1-YR RETURN
# --------------------------------
# Features: include Sharpe Ratio and categorical variables
X = df[[sharpe_col, market_col, type_col, risk_col]]
y = df[return_1yr_col]

# Preprocessing pipelines
```

```
numeric_features = [sharpe_col]
categorical_features = [market_col, type_col, risk_col]

numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])

# Random Forest Regressor pipeline
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(random_state=42, n_estimators=200))
])

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)
print(f"\n1-Year Return Prediction - Test R2: {r2_score(y_test, y_pred):.4f},
MAE: {mean_absolute_error(y_test, y_pred):.4f}")

# -------------------------------
# Find optimal Sharpe Ratio
# -------------------------------
# Create baseline (median numeric, mode categorical)
baseline = {}
baseline[sharpe_col] = 0  # placeholder, will vary
for col in categorical_features:
    baseline[col] = X[col].mode()[0]

# Generate a Sharpe ratio grid
sharpe_min, sharpe_max = X[sharpe_col].min(), X[sharpe_col].max()
sharpe_grid = np.linspace(sharpe_min, sharpe_max, 200)
pred_returns = []

for s in sharpe_grid:
    row = baseline.copy()
    row[sharpe_col] = s
    row_df = pd.DataFrame([row])
    pred = model.predict(row_df)[0]
    pred_returns.append(pred)

pred_returns = np.array(pred_returns)
best_idx = pred_returns.argmax()
optimal_sharpe = sharpe_grid[best_idx]
optimal_return = pred_returns[best_idx]

# Visualization
plt.figure(figsize=(10,6))
plt.plot(sharpe_grid, pred_returns, label='Predicted 1YrReturn')
plt.axvline(optimal_sharpe, color='r', linestyle='--', label=f'Optimal Sharpe ≈
{optimal_sharpe:.3f}')
plt.xlabel("Sharpe Ratio")
```

```
plt.ylabel("Predicted 1-Year Return (%)")
plt.title("Predicted 1-Year Return vs Sharpe Ratio")
plt.legend()
plt.tight_layout()
plt.savefig("sharpe_vs_1yrreturn.png", dpi=150)
plt.close()

print(f"\n  Optimal Sharpe Ratio to maximize 1-Year Return:
{optimal_sharpe:.4f}")
print(f"Predicted 1-Year Return at optimal Sharpe: {optimal_return:.4f}%")

# -------------------------------
# ANALYSIS SUMMARY
# -------------------------------
print("\n  Analysis Summary:")
print(f"- Best MarketCap|Type|Risk combination (3YrReturn):
{best_combo[market_col]} | {best_combo[type_col]} | {best_combo[risk_col]}")
print(f"- Predicted 3YrReturn: {best_combo[return_3yr_col]:.2f}%")
print(f"- Optimal Sharpe Ratio to maximize 1YrReturn: {optimal_sharpe:.4f}")
print(f"- Predicted 1YrReturn at optimal Sharpe: {optimal_return:.2f}%")
```