MAULANA AZAD
NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL – 462003 (INDIA)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BUDGET MANAGEMENT ANALYSIS

AI PROJECT REPORT

Submitted by:

Vartika Koli	**201112047**
Charu Agarwal	**201112048**

Under the Guidance of
Dr. Manali
Session: 2022-2023

ABSTRACT

Budget management is a critical aspect of personal financial planning and involves the effective planning, tracking, and control of expenses to ensure financial stability and achieve financial goals. Traditional budget management methods may require manual data entry, calculation, and analysis, which can be time-consuming and prone to errors. With the advent of artificial intelligence (AI) technologies, budget management can be enhanced through automated and data-driven approaches. This abstract presents an overview of how AI can be leveraged for budget management, including techniques such as predictive analytics, machine learning, and natural language processing. These technologies can help individuals in budget creation, expense categorization, spending analysis, and financial goal tracking. Furthermore, AI-powered budget management tools can provide personalized recommendations and insights to optimize spending, increase savings, and improve financial decision-making. The abstract also discusses the benefits and challenges of using AI for budget management, including data privacy, security, and ethical considerations.

AI-powered budget management tools can integrate with other financial management tools, such as mobile apps, accounting software, and banking platforms, to provide a holistic view of an individual's financial health. AI can also enable personalized budgeting plans, taking into consideration an individual's financial goals, income, and spending habits, and dynamically adjusting the budget plan as needed.

Overall, AI has the potential to revolutionize budget management by offering advanced capabilities for data-driven analysis, decision-making, and optimization, empowering individuals to effectively manage their finances and achieve their financial goals.

INTRODUCTION

Budget management refers to the process of planning, organizing, and controlling one's expenses and income to achieve financial goals. It involves creating a plan that outlines expected income, expenses, savings, and investments, and then tracking and managing these financial activities to ensure they align with the established budget.

1. **Background**
   The background of budget management includes considerations of data privacy, security, and ethical use of AI. Proper measures need to be in place to ensure the protection of personal and financial data and compliance with applicable laws and regulations. Ethical considerations, such as transparency, fairness, and accountability, are also important in the development and use of AI-powered budget management tools.
2. **Motivation**
   Traditional budget management methods often involve manual tracking of expenses, income, and savings using spreadsheets or pen-and-paper methods. However, with the advancements in technology, including the advent of artificial intelligence (AI), budget management can now be automated and optimized with sophisticated algorithms, machine learning, and data-driven analysis.
3. **Objectives**
   The objective of this code is to analyse and categorize our expenses which is stored in a CSV file. It performs various tasks such as categorizing your expenses and predict the upcoming expenses and suggest the investment options according the input given by user including expenses and monthly income.

4. **Expected Outcomes**
   1. Group expenses by category and calculate the total for each category.
   2. Predict the expenses according to the historical data
   3. Visualise the expenses
   4. Provide the suggested mode of investments and returns according to the provided data

5. **Significance of Project**
   The significance of a budget management project lies in its potential to positively impact an individual's financial health and overall financial well-being. Here are some key aspects of the significance of a budget management project:

1. Financial discipline: Budget management helps individuals develop and maintain financial discipline by setting spending limits, tracking expenses, and ensuring that expenses do not exceed income. This promotes responsible financial habits, such as saving, investing, and avoiding unnecessary debt, leading to improved financial stability.
2. Goal achievement: Budget management allows individuals to set and prioritize financial goals, such as saving for emergencies, retirement, education, or other life events. A well-managed budget ensures that funds are allocated towards these goals, helping individuals make progress towards achieving them.
3. Financial awareness: Budget management encourages individuals to be aware of their income, expenses, and overall financial situation. It provides insights into spending patterns, areas of overspending or potential savings, and helps individuals make informed financial decisions based on their financial status.
4. Debt management: Budget management aids in managing and reducing debt by tracking debt payments, avoiding unnecessary debt, and allocating funds towards debt repayments. This can help individuals pay off debts faster and reduce financial stress.
5. Investment management: Budget management can include provisions for investment planning and tracking, helping individuals optimize their investment strategies, and potentially increase their wealth over time.
6. Financial decision-making: Budget management provides a framework for making informed financial decisions based on available funds and financial goals. It helps individuals prioritize expenses, allocate resources, and make financial choices aligned with their long-term financial objectives.
7. Financial security: Budget management promotes financial security by ensuring that individuals have a clear understanding of their financial situation, are prepared for emergencies, and have a plan to achieve their financial goals. This can provide peace of mind and reduce financial stress.

In summary, the significance of a budget management project lies in its ability to promote financial discipline, goal achievement, financial awareness, debt management, investment management, informed decision-making, and financial security. It can help individuals improve their financial health, achieve their financial goals, and enhance their overall financial well-being.

IMPLEMENTATION METHODOLOGY

The methodology used in this project can be summed up in the following steps:

1. **Dataset Preparation**
    - The dataset in form of csv file is given in zip file
    - The dataset we are using here is our transaction history which includes 40-50 transactions in a csv file
    - It contains columns named “description, amount, date, category”
    - Date is in the format '%d-%m-%Y'
    - The monthly income used for investments and returns is too provided in zip file.

2. **Data Pre-Processing**
    - Convert the date column to a datetime object
    - Convert the amount column to a float
    - Drop any rows with missing values and remove any leading/trailing whitespace from category name
    - Group expenses by category and calculate the total for each category
    - Split the data into training and test sets
    - Vectorize the textual descriptions using a TfidfVectorizer

3. **Model Architecture:**
    - We are training an SVM classifier on the vectorized textual descriptions of the training data, and then using the trained classifier to predict the categories for the test data.

- The LinearSVC class is specifically designed for binary classification with linear kernels, meaning it can be used to train a binary classifier that separates data points into two classes based on a linear decision boundary.
- The fit method uses the training data to learn the parameters of the SVM model and find the optimal decision boundary that separates the data into the specified categories.

4. **Natural Language Processing:**
    - Loads a pre-trained English language model from spaCy using the 'en_core_web_sm' package.
    - Reads expense data from a CSV file into a Pandas dataframe (df).
    - Converts the dataframe (df) to a list of Expense objects, where each row in the dataframe is converted to an Expense object using the defined Expense class and appended to the list of expenses.
    - Defines a function (extract_category) that takes an expense description as input and uses spaCy to extract category information from the description. The function looks for keywords in the noun chunks of the description to determine the category, and returns the corresponding category as a string.
    - Iterates over the list of expenses and calls the extract_category function on each expense to extract the category information and update the category field of the Expense object.
    - Converts the list of Expense objects to a list of dictionaries (expense_dicts), where each Expense object is converted to a dictionary with keys for date, description, amount, and category.
    - Converts the list of dictionaries (expense_dicts) to a JSON string (json_str) using the json.dumps() function.
    - Reads the JSON string (json_str) back into a Pandas dataframe (df) using the pd.read_json() function.
    - Prints the updated dataframe (df), which now contains the extracted category information for each expense.

5. Prediction

- Group expenses by category and calculate the total for each category.
- Predict the expenses according to the historical data
- Provide the suggested mode of investments and returns according to the provided data

6. Data Visualization

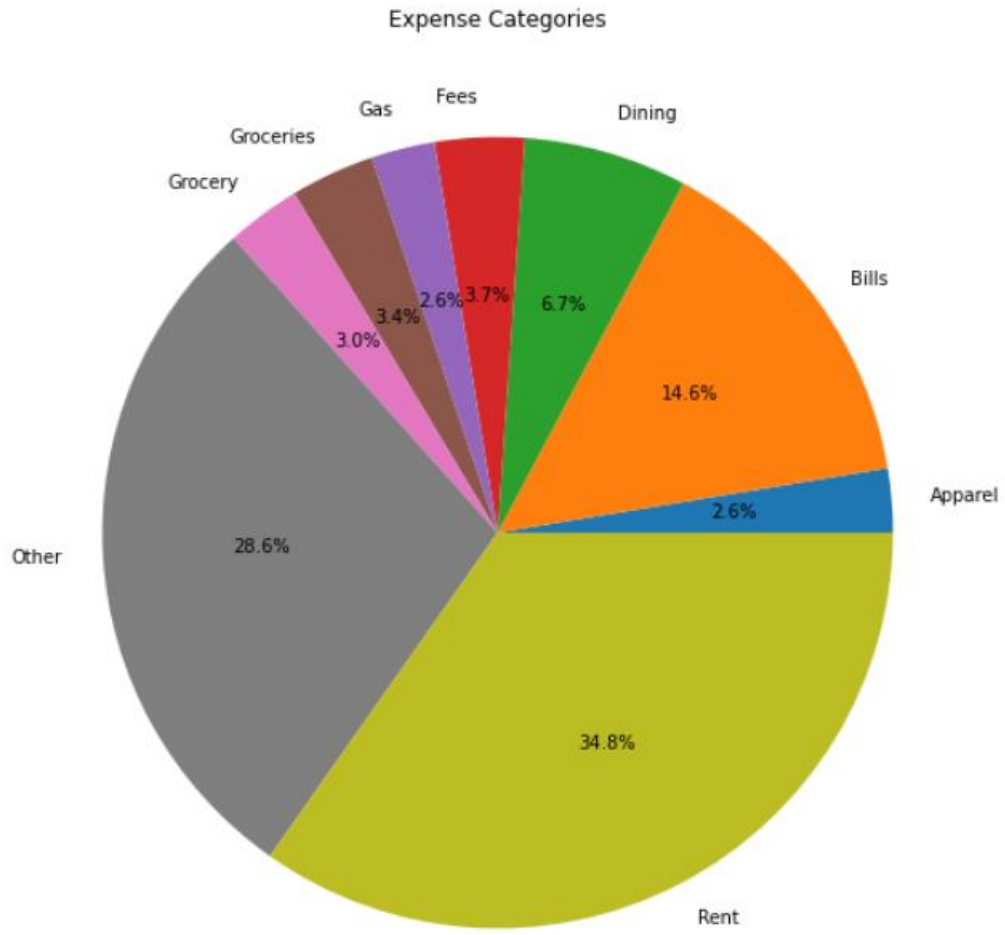
Visualize the predicted data using Charts and Graphs

RESULTS

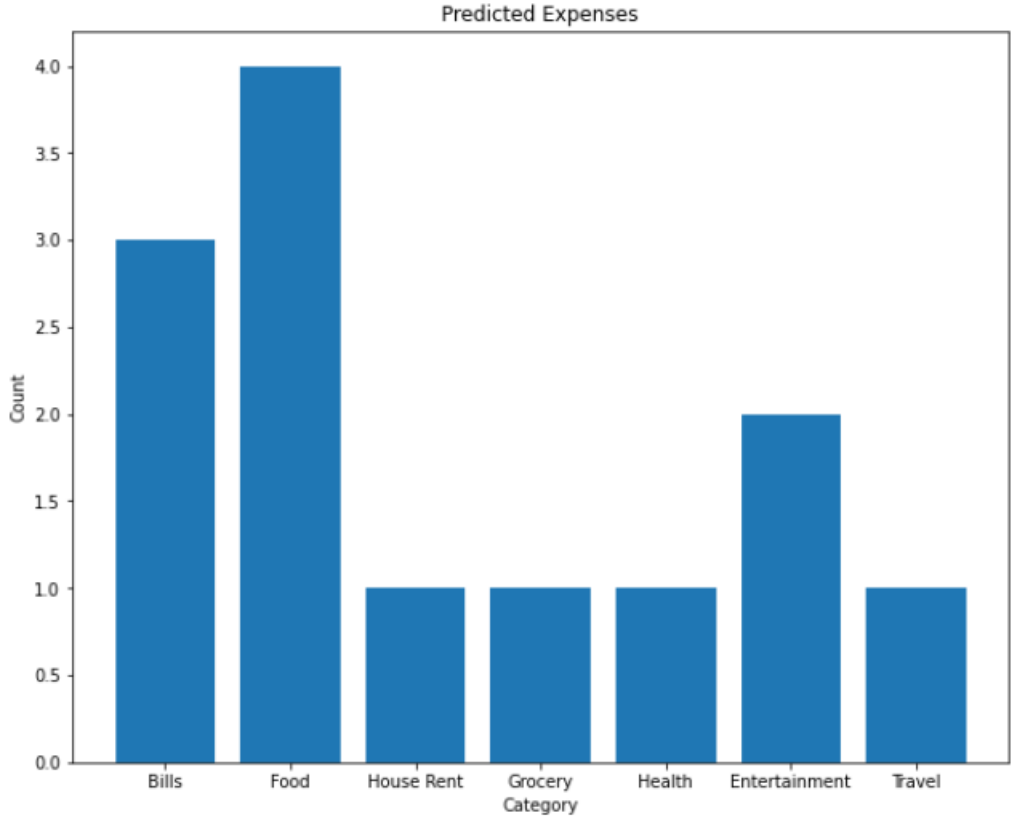
Group expenses by category and calculate the total for each category

```
category
Bills                          740.0
Clothing                       235.0
Electricity and water bills    100.0
Entertainment                   59.5
Fee                            210.0
Food                           482.9
Gifts                           50.0
Grocery                        275.0
Health                         100.0
House Rent                    2000.0
Miscallenous                    75.0
Personal care                   25.0
Transportation                 750.0
Travel                         650.0
Name: amount, dtype: float64
```

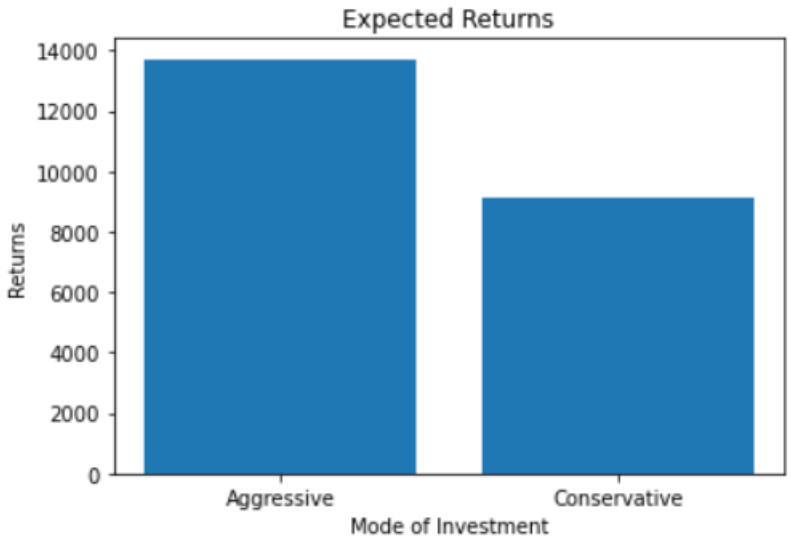
Visualize the Grouped expenses



Predict the coming expenses categorically



Determine the mode of investment that maximizes returns




```
The mode of investment that maximizes returns is: Conservative
The expected returns for the year are: 9139.808
```

CODE:

Dataset Preparation

```
import pandas as pd

# Load the CSV file into a Pandas dataframe
df = pd.read_csv('expenses.csv')
# Convert the date column to a datetime object
df['date'] = pd.to_datetime(df['date'], format='%d-%m-%Y')
# Convert the amount column to a float
df['amount'] = df['amount'].astype(float)

# Drop any rows with missing values
df = df.dropna()
```

Group Expenses

```
# Remove any leading/trailing whitespace from category names
df['category'] = df['category'].str.strip()

# Group expenses by category and calculate the total for each
category
category_totals = df.groupby('category')['amount'].sum()

# Print the category totals
print(category_totals)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score

# Load the CSV file into a Pandas dataframe
df = pd.read_csv('expenses.csv')
```

Data Pre-Processing

```
# Convert the date column to a datetime object
df['date'] = pd.to_datetime(df['date'], format='%d-%m-%Y')

# Convert the amount column to a float
df['amount'] = df['amount'].astype(float)

# Drop any rows with missing values
df = df.dropna()

# Remove any leading/trailing whitespace from category names
df['category'] = df['category'].str.strip()

# Split the data into training and test sets
```

```
X_train, X_test, y_train, y_test =
train_test_split(df['description'], df['category'], random_state=42)

# Vectorize the textual descriptions using a TfidfVectorizer
vectorizer = TfidfVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)
```

Model Architecture:

```
# Train an SVM classifier on the training data
clf = LinearSVC()
clf.fit(X_train_counts, y_train)

# Predict the categories for the test data
y_pred = clf.predict(X_test_counts)

# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

import pandas as pd
import json
import spacy
```

Natural Language Processing

```
# Load the pre-trained English language model in spaCy
nlp = spacy.load('en_core_web_sm')

# Define an Expense class with fields for date, description, amount,
and category
class Expense:
    def __init__(self, date, description, amount, category):
        self.date = date
        self.description = description
        self.amount = amount
        self.category = category

# Load the expense data from a CSV file into a Pandas dataframe
df = pd.read_csv('expenses.csv')

# Convert the dataframe to a list of Expense objects
expenses = [Expense(pd.to_datetime(row['date'], format='%d-%m-
%Y').strftime('%Y-%m-%d'), row['description'], row['amount'], '') for
_, row in df.iterrows()]

# Define a function to extract category information from expense
descriptions
def extract_category(description: str) -> str:
    # Create a spaCy Doc object from the expense description
    doc = nlp(description.lower())

    # Extract the noun chunks from the Doc object
    noun_chunks = list(doc.noun_chunks)

    # Look for keywords in the noun chunks to determine the category
```

```
    if any(['coffee' in nc.text for nc in noun_chunks]):
        return 'Coffee'
    elif any(['groceries' in nc.text for nc in noun_chunks]):
        return 'Groceries'
    elif any(['dinner' in nc.text for nc in noun_chunks]):
        return 'Dining'
    elif any(['lunch' in nc.text for nc in noun_chunks]):
        return 'Dining'
    elif any(['gas' in nc.text for nc in noun_chunks]):
        return 'Gas'
    elif any(['movie tickets' in nc.text for nc in noun_chunks]):
        return 'Entertainment'
    elif any(['hotel stay' in nc.text for nc in noun_chunks]):
        return 'Lodging'
    elif any(['clothing' in nc.text for nc in noun_chunks]):
        return 'Apparel'
    elif any(['bill' in nc.text for nc in noun_chunks]):
        return 'Bills'
    elif any(['fee' in nc.text for nc in noun_chunks]):
        return 'Fees'
    elif any(['rent' in nc.text for nc in noun_chunks]):
        return 'Rent'
    elif any(['grocery' in nc.text for nc in noun_chunks]):
        return 'Grocery'
    else:
        return 'Other'

# Extract category information from the expense descriptions
for expense in expenses:
    expense.category = extract_category(expense.description)

# Convert the list of Expense objects to a list of dictionaries
expense_dicts = [{'date': e.date, 'description': e.description,
'amount': e.amount, 'category': e.category} for e in expenses]

# Convert the list of dictionaries to a JSON string
json_str = json.dumps(expense_dicts)

# Convert the JSON string back to a Pandas dataframe
df = pd.read_json(json_str)

# Print the updated dataframe
print(df)

import pandas as pd
import spacy
import matplotlib.pyplot as plt

# Define an Expense class with fields for date, description, amount,
and category
class Expense:
    def __init__(self, date, description, amount, category):
        self.date = date
        self.description = description
        self.amount = amount
        self.category = category
# Load the pre-trained English language model in spaCy
```

```
nlp = spacy.load('en_core_web_sm')
# Load the expense data from a CSV file into a Pandas dataframe
df = pd.read_csv('expenses.csv')

# Convert the dataframe to a list of Expense objects
expenses = [Expense(row['date'], row['description'], row['amount'],
'') for _, row in df.iterrows()]

# Define a function to extract category information from expense
descriptions
def extract_category(description: str) -> str:
    # Create a spaCy Doc object from the expense description
    doc = nlp(description.lower())

    # Extract the noun chunks from the Doc object
    noun_chunks = list(doc.noun_chunks)

    # Look for keywords in the noun chunks to determine the category
    if any(['coffee' in nc.text for nc in noun_chunks]):
        return 'Coffee'
    elif any(['groceries' in nc.text for nc in noun_chunks]):
        return 'Groceries'
    elif any(['dinner' in nc.text for nc in noun_chunks]):
        return 'Dining'
    elif any(['lunch' in nc.text for nc in noun_chunks]):
        return 'Dining'
    elif any(['gas' in nc.text for nc in noun_chunks]):
        return 'Gas'
    elif any(['movie tickets' in nc.text for nc in noun_chunks]):
        return 'Entertainment'
    elif any(['hotel stay' in nc.text for nc in noun_chunks]):
        return 'Lodging'
    elif any(['clothing' in nc.text for nc in noun_chunks]):
        return 'Apparel'
    elif any(['bill' in nc.text for nc in noun_chunks]):
        return 'Bills'
    elif any(['fee' in nc.text for nc in noun_chunks]):
        return 'Fees'
    elif any(['rent' in nc.text for nc in noun_chunks]):
        return 'Rent'
    elif any(['grocery' in nc.text for nc in noun_chunks]):
        return 'Grocery'
    else:
        return 'Other'



# Extract category information from the expense descriptions
for expense in expenses:
    expense.category = extract_category(expense.description)

# Convert the list of Expense objects to a Pandas dataframe
df = pd.DataFrame.from_records([e.__dict__ for e in expenses])
```

Visualise Grouped Expenses

```
# Group the expenses by category and sum the amounts for each
category
```

```
grouped = df.groupby('category').agg({'amount': 'sum'})

# Create a pie chart showing the distribution of expenses by category
fig, ax = plt.subplots(figsize=(10,10))
ax.pie(grouped['amount'], labels=grouped.index, autopct='%1.1f%%')
ax.set_title('Expense Categories')

# Show the pie chart
plt.show()

import matplotlib.pyplot as plt
```

Predict Upcoming Expenses

```
# Create a dictionary to store the count of each predicted category
pred_counts = {}
for category in y_pred:
    if category in pred_counts:
        pred_counts[category] += 1
    else:
        pred_counts[category] = 1

# Create a bar plot showing the count of each predicted category
fig, ax = plt.subplots(figsize=(10, 8))
ax.bar(pred_counts.keys(), pred_counts.values())
ax.set_title('Predicted Expenses')
ax.set_xlabel('Category')
ax.set_ylabel('Count')
plt.show()

import pandas as pd

# read in the expenses DataFrame
expenses_df = pd.read_csv("expenses.csv")

# convert the date column to a pandas datetime object
expenses_df["date"] = pd.to_datetime(expenses_df["date"], format="%d-
%m-%Y")
```

Take Monthly income and Calculate Savings

```
# read in the monthly income DataFrame
monthly_df = pd.read_csv("monthly_income.csv")

# add a new column to the expenses DataFrame for the month
expenses_df["month"] = expenses_df["date"].dt.month

# group the expenses DataFrame by month and category, and sum the
amounts
expenses_grouped = expenses_df.groupby(["month",
"category"])["amount"].sum().reset_index()

# pivot the expenses DataFrame to get the categories as columns
expenses_pivot = expenses_grouped.pivot(index="month",
columns="category", values="amount").reset_index()

# merge the expenses and monthly income DataFrames on the month
column
```

```
result_df = pd.merge(monthly_df, expenses_pivot, on="month",
how="outer")

# fill any missing values with 0
result_df.fillna(0, inplace=True)

# add a new column for the total expenses for each month
result_df["expense"] = result_df.drop("month", axis=1).sum(axis=1)

# reorder the columns
result_df = result_df[["month", "expense", "income"]]

# save the result DataFrame to a new CSV file
result_df.to_csv("monthly_income_and_expenses.csv", index=False)

#To visualize the expected returns, we can add a bar chart using the
matplotlib library. Here's the modified code:
```

Mode of Investment and Returns

```
import pandas as pd
import matplotlib.pyplot as plt

#Load data from CSV file
data = pd.read_csv('monthly_income_and_expenses.csv')

#Define the expected returns for each mode of investment
aggressive_returns = 0.12
conservative_returns = 0.08

#Calculate net income (income - expense) for each month
data['Net Income'] = data['income'] - data['expense']

#Calculate the percentage of net income relative to income for each
month
data['Net Income %'] = (data['Net Income'] / data['income']) * 100

#Calculate the rolling sum of net income percentage for each month
data['Cumulative Net Income %'] = data['Net Income %'].cumsum()

#Find the index of the month with the highest cumulative net income
percentage
max_index = data['Cumulative Net Income %'].idxmax()

#Determine the mode of investment that maximizes returns
if max_index < 6: # First half of the year
    mode_of_investment = 'Aggressive'
    expected_returns = aggressive_returns
else: # Second half of the year
    mode_of_investment = 'Conservative'
    expected_returns = conservative_returns

#Calculate the total expected returns for the year
total_returns = data['Net Income'].sum() * expected_returns

#Plot a bar chart of the expected returns
plt.bar(['Aggressive', 'Conservative'], [data['Net Income'].sum() *
aggressive_returns, data['Net Income'].sum() * conservative_returns])
plt.title('Expected Returns')
```

```
plt.xlabel('Mode of Investment')
plt.ylabel('Returns')
plt.show()

print("The mode of investment that maximizes returns is:",
mode_of_investment)
print("The expected returns for the year are:", total_returns)
```