(SEC1) 4 - Classwork

February 1, 2024

1 HR ATTRIBUTION

[1]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, f1_score
import numpy as np
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, auc
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, roc_auc_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score
```

2 1.) Import, split data into X/y, plot y data as bar charts, turn X categorical variables binary and tts.

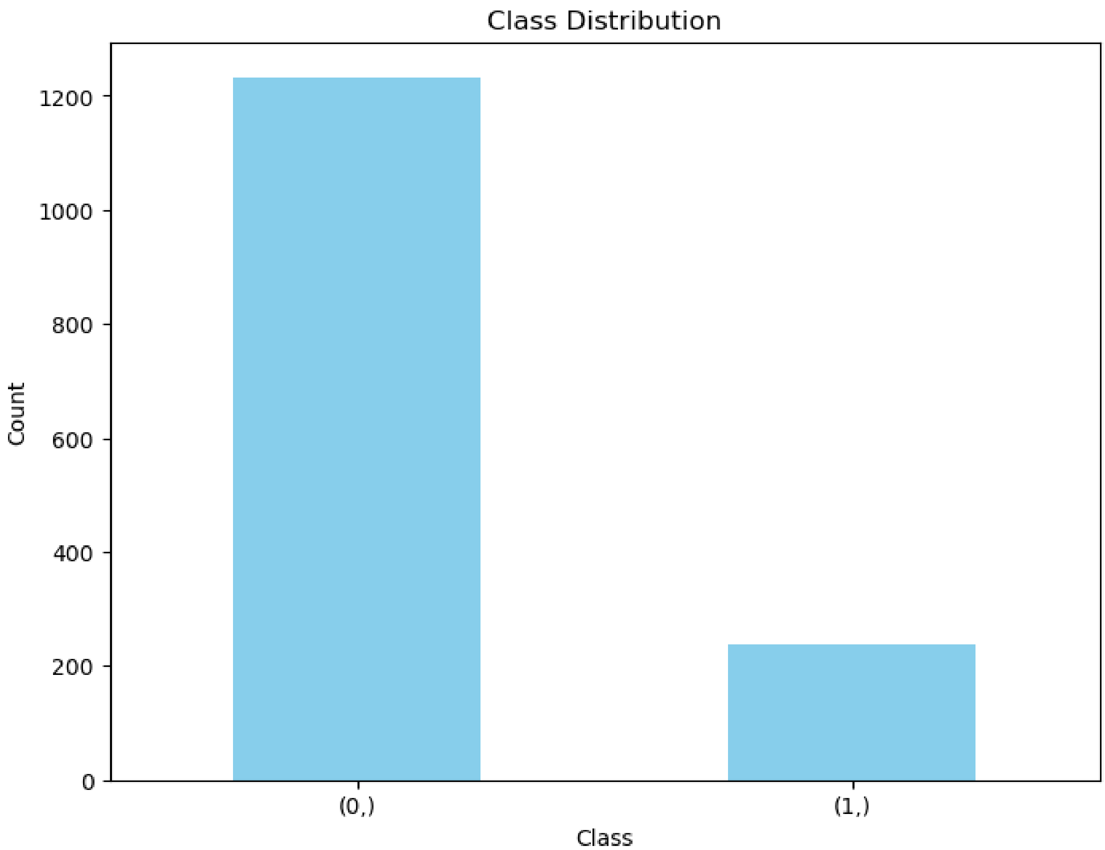
[2]:
```
df = pd.read_csv("HR_Analytics.csv")
```

[3]:
```
y = df[["Attrition"]].copy()
X = df.drop("Attrition", axis = 1)
```

[4]:
```
y["Attrition"] = [1 if i == "Yes" else 0 for i in y["Attrition"]]
```

[5]:
```
class_counts = y.value_counts()


plt.figure(figsize=(8, 6))
class_counts.plot(kind='bar', color='skyblue')
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution')
```

```
plt.xticks(rotation=0)  # Remove rotation of x-axis labels
plt.show()
```




```
[6]: # Step 1: Identify string columns
string_columns = X.columns[X.dtypes == 'object']

# Step 2: Convert string columns to categorical
for col in string_columns:
    X[col] = pd.Categorical(X[col])

# Step 3: Create dummy columns
X = pd.get_dummies(X, columns=string_columns,␣
 ↪prefix=string_columns,drop_first=True)
```

```
[7]: x_train,x_test,y_train,y_test=train_test_split(X,
 y, test_size=0.20, random_state=42)
```

3 2.) Using the default Decision Tree. What is the IN/Out of Sample accuracy?

[11]:
```
clf = DecisionTreeClassifier()
clf.fit(x_train,y_train)
y_pred=clf.predict(x_train)
acc=accuracy_score(y_train,y_pred)
print("IN SAMPLE ACCURACY : " , round(acc,2))

y_pred=clf.predict(x_test)
acc=accuracy_score(y_test,y_pred)
print("OUT OF SAMPLE ACCURACY : " , round(acc,2))
```

```
IN SAMPLE ACCURACY :  1.0
OUT OF SAMPLE ACCURACY :  0.78
```

4 3.) Run a grid search cross validation using F1 score to find the best metrics. What is the In and Out of Sample now?

[9]:
```
# Define the hyperparameter grid to search through
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': np.arange(1, 11),  # Range of max_depth values to try
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}


dt_classifier = DecisionTreeClassifier(random_state=42)

scoring = make_scorer(f1_score, average='weighted')

grid_search = GridSearchCV(estimator=dt_classifier, param_grid=param_grid,␣
 ↪scoring=scoring, cv=5)

grid_search.fit(x_train, y_train)

# Get the best parameters and the best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best Parameters:", best_params)
print("Best F1-Score:", best_score)
```

```
Best Parameters: {'criterion': 'gini', 'max_depth': 6, 'min_samples_leaf': 2,
'min_samples_split': 2}
Best F1-Score: 0.8214764475510983
```

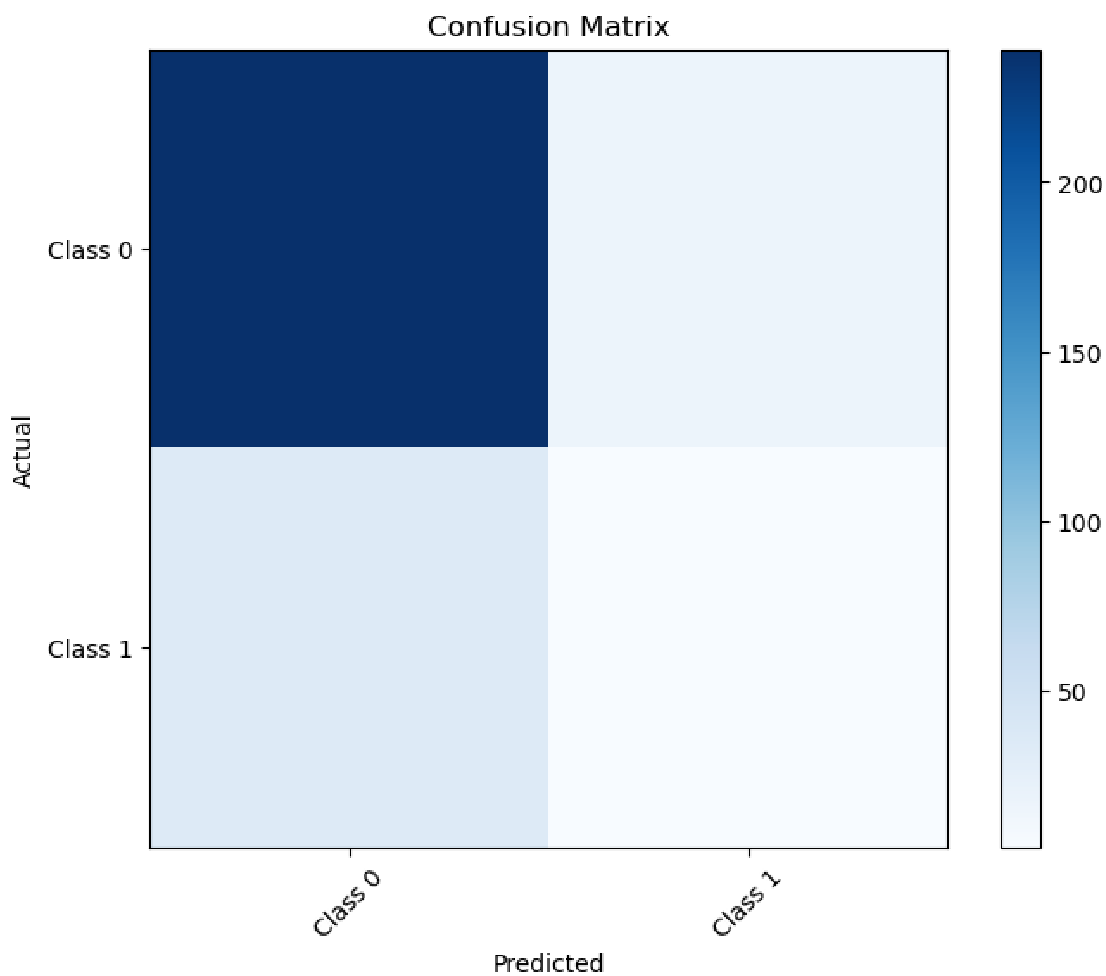
[12]:
```
clf = tree.DecisionTreeClassifier(**best_params, random_state =42)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_train)
acc=accuracy_score(y_train,y_pred)
print("IN SAMPLE ACCURACY : " , round(acc,2))

y_pred=clf.predict(x_test)
acc=accuracy_score(y_test,y_pred)
print("OUT OF SAMPLE ACCURACY : " , round(acc,2))
```

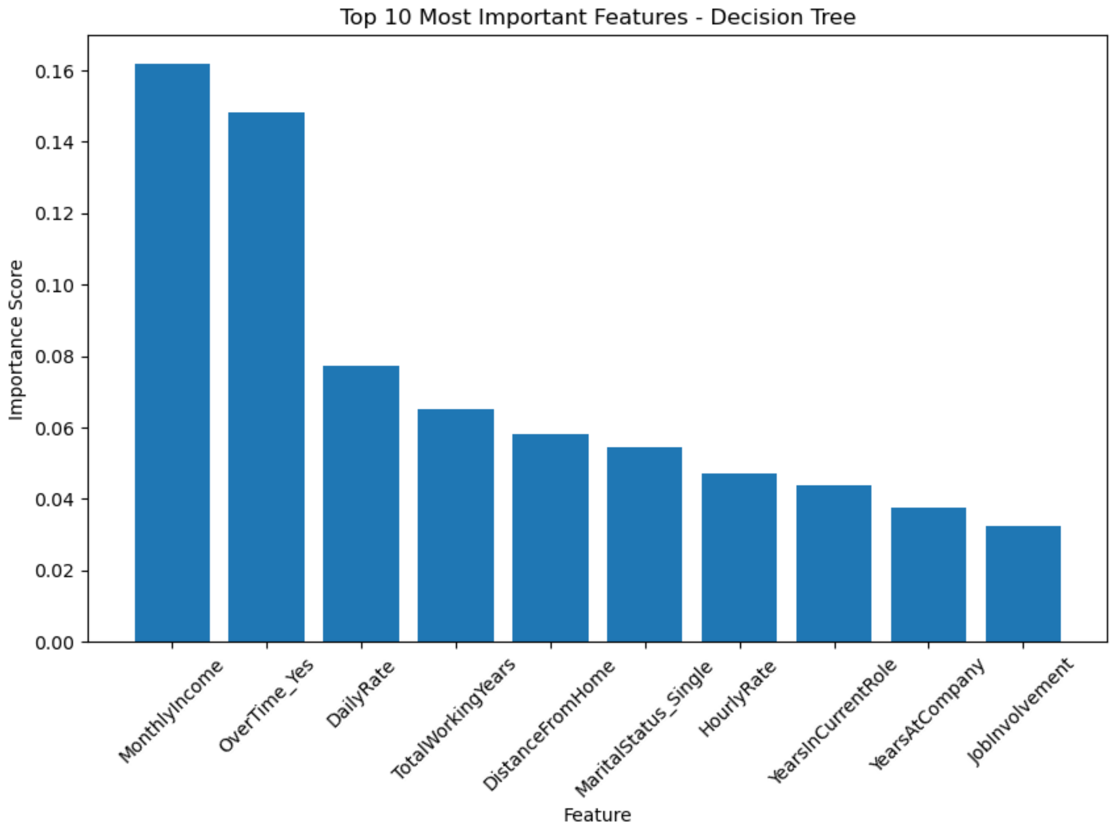
```
IN SAMPLE ACCURACY :  0.91
OUT OF SAMPLE ACCURACY :  0.83
```

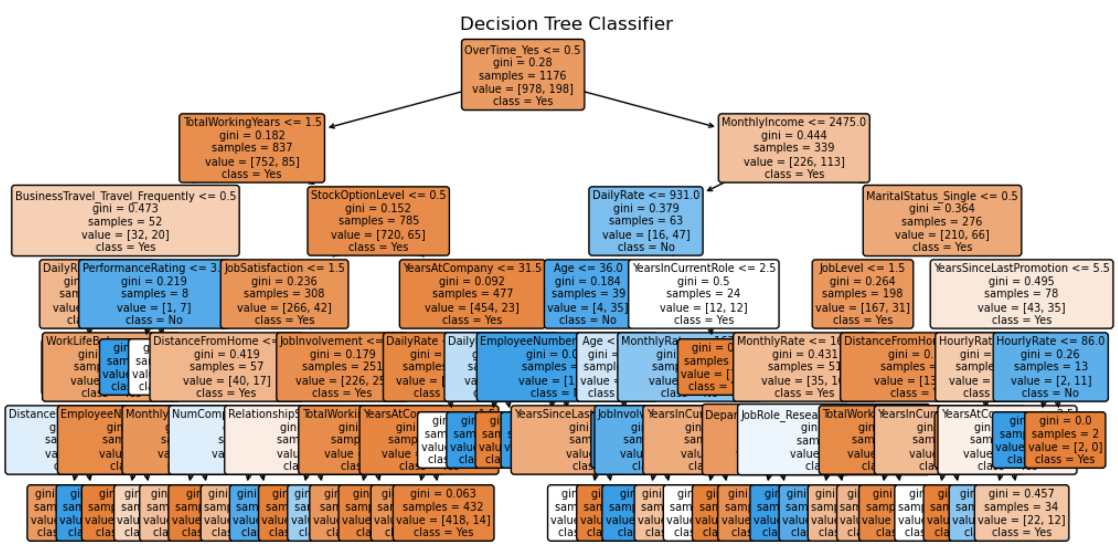
5 4.) Plot ……

[13]:
```
# Make predictions on the test data
y_pred = clf.predict(x_test)
y_prob = clf.predict_proba(x_test)[:, 1]

# Calculate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
plt.imshow(conf_matrix, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(conf_matrix))
plt.xticks(tick_marks, ['Class 0', 'Class 1'], rotation=45)
plt.yticks(tick_marks, ['Class 0', 'Class 1'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()




feature_importance = clf.feature_importances_

# Sort features by importance and select the top 10
top_n = 10
top_feature_indices = np.argsort(feature_importance)[::-1][:top_n]
top_feature_names = X.columns[top_feature_indices]
top_feature_importance = feature_importance[top_feature_indices]

# Plot the top 10 most important features
plt.figure(figsize=(10, 6))
```

```python
plt.bar(top_feature_names, top_feature_importance)
plt.xlabel('Feature')
plt.ylabel('Importance Score')
plt.title('Top 10 Most Important Features - Decision Tree')
plt.xticks(rotation=45)
plt.show()

# Plot the Decision Tree for better visualization of the selected features
plt.figure(figsize=(12, 6))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=["Yes", "No"],
 ↪rounded=True, fontsize=7)
plt.title('Decision Tree Classifier')
plt.show()
```




Top 10 Most Important Features - Decision Tree
Importance Score
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
MonthlyIncome
OverTime_Yes
DailyRate
TotalWorkingYears
DistanceFromHome
MaritalStatus_Single
HourlyRate
YearsInCurrentRole
YearsAtCompany
JobInvolvement
Feature


Decision Tree Classifier
OverTime_Yes <= 0.5
gini = 0.28
samples = 1176
value = [978, 198]
class = Yes
TotalWorkingYears <= 1.5
gini = 0.182
samples = 837
value = [752, 85]
class = Yes
MonthlyIncome <= 2475.0
gini = 0.444
samples = 339
value = [226, 113]
class = Yes
BusinessTravel_Travel_Frequently <= 0.5
gini = 0.473
samples = 52
value = [32, 20]
class = Yes
StockOptionLevel <= 0.5
gini = 0.152
samples = 785
value = [720, 65]
class = Yes
DailyRate <= 931.0
gini = 0.379
samples = 63
value = [16, 47]
class = No
MaritalStatus_Single <= 0.5
gini = 0.364
samples = 276
value = [210, 66]
class = Yes

[ ]:

[ ]:

```

```

6 5.) Looking at the graphs. what would be your suggestions to try to improve employee retention? What additional information would you need for a better plan. Plot anything you think would assist in your assessment.

6.1 ANSWER :

[16]:

```
np.corrcoef(np.array(X['OverTime_Yes']),np.array(y["Attrition"]))
```

[16]:

```
array([[1.        , 0.24611799],
       [0.24611799, 1.        ]])
```

[ ]:

```

```

[ ]:

```

```

7 6.) Using the Training Data, if they made everyone work overtime. What would have been the expected difference in employee retention?

[17]:

```
x_train_experiment = x_train.copy()
```

[18]:

```
x_train_experiment["OverTime_Yes"] = 0
```

[19]:

```
y_pred = clf.predict(x_train)
y_pred_experiment = clf.predict(x_train_experiment)
```

[20]:

```
diff = sum(y_pred-y_pred_experiment)
diff
```

[20]:

```
59
```

8 7.) If they company loses an employee, there is a cost to train a new employee for a role ~2.8 * their monthly income.

9 To make someone not work overtime costs the company 2K per person.

10 Is it profitable for the company to remove overtime? If so/not by how much?

11 What do you suggest to maximize company profits?

[26]:
```
x_train_experiment["Y"] = y_pred
x_train_experiment["Y_exp"] = y_pred_experiment
x_train_experiment["Ret_Change"] = x_train_experiment["Y_exp"] -␣
↪x_train_experiment["Y"]
```

[27]:
```
# Savings
sav = sum(-2.8 *␣
↪x_train_experiment['Ret_Change']*x_train_experiment["MonthlyIncome"])
```

[28]:
```
x_train_experiment["Ret_Change"]
```

[28]:
```
1097    0
727     0
254     0
1175    0
1341    0
       ..
1130    0
1294    0
860     1
1459    0
1126    0
Name: Ret_Change, Length: 1176, dtype: int64
```

[32]:
```
cost = len(x_train[x_train["OverTime_Yes"]==1])*2000
sav-cost
```

[32]:
```
-117593.99999999977
```

11.1 ANSWER :

12 8.) Use your model and get the expected change in retention for raising and lowering peoples income. Plot the outcome of the experiment. Comment on the outcome of the experiment and your suggestions to maximize profit.

[ ]:
```
raise = 500
```

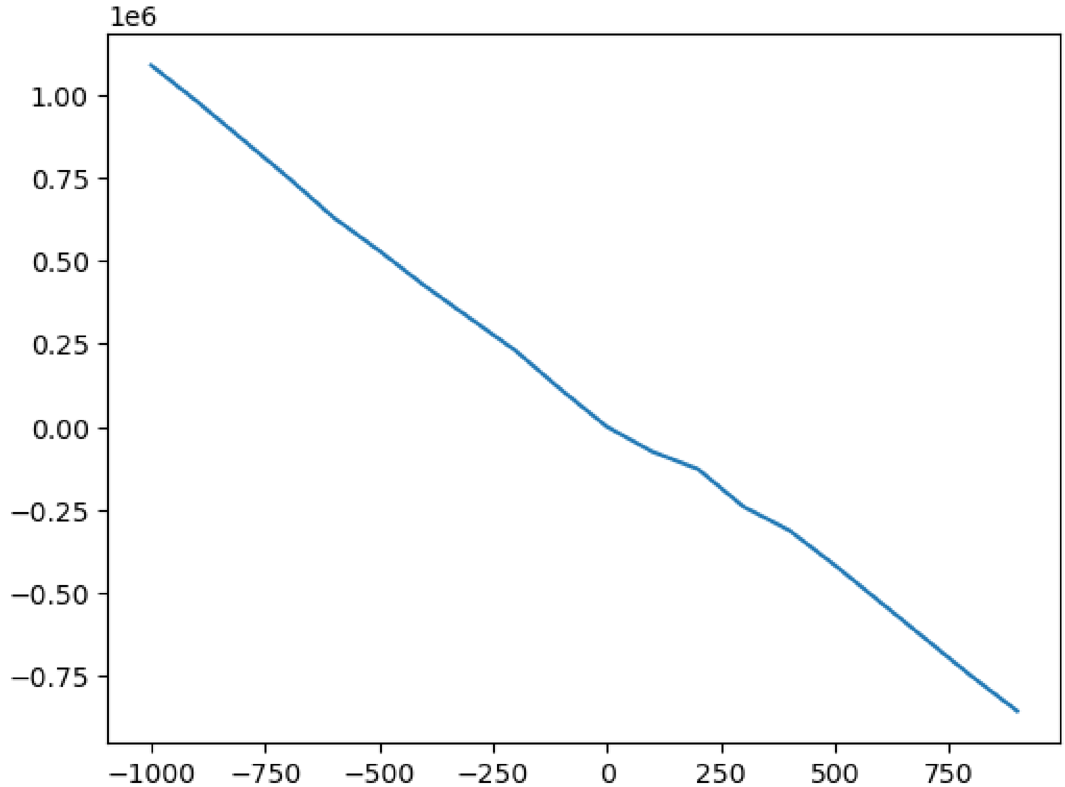
[33]:
```
profits = []
for raise_amount in range(-1000, 1000,100):
    x_train_experiment= x_train.copy()
    x_train_experiment ["MonthlyIncome"] = x_train_experiment␣
 ↪["MonthlyIncome"]+ raise_amount

    y_pred = clf.predict(x_train)
    y_pred_experiment = clf.predict(x_train_experiment)

    diff = sum(y_pred - y_pred_experiment)
    print ("Change in attrition", diff)
    x_train_experiment ["Y"] = y_pred
    x_train_experiment ["Y_exp"] = y_pred_experiment

    x_train_experiment["RetChange"] = x_train_experiment ["Y_exp"] -␣
 ↪x_train_experiment["Y"]
    sav = sum(-2.
 ↪8*x_train_experiment["RetChange"]*x_train_experiment["MonthlyIncome"])
    cost = len(x_train)*raise_amount
    print("profits,", sav-cost)
    profits.append(sav-cost)
```

```
Change in attrition -16
profits, 1087584.4
Change in attrition -14
profits, 979524.0
Change in attrition -13
profits, 864992.8
Change in attrition -12
profits, 750738.8
Change in attrition -12
profits, 629778.8
Change in attrition -9
profits, 530138.0
Change in attrition -7
profits, 424200.0
Change in attrition -4
profits, 326096.4
```

```
Change in attrition -1
profits, 228440.8
Change in attrition -1
profits, 110714.8
Change in attrition 0
profits, 0.0
Change in attrition 6
profits, -75328.40000000001
Change in attrition 15
profits, -127503.60000000002
Change in attrition 15
profits, -240914.8
Change in attrition 21
profits, -311586.80000000005
Change in attrition 22
profits, -416449.6000000001
Change in attrition 22
profits, -527889.6000000001
Change in attrition 22
profits, -639329.6000000001
Change in attrition 22
profits, -750769.6000000001
Change in attrition 23
profits, -854999.6000000001
```

12.1 ANSWER :

[34]:
```
plt.plot(range(-1000, 1000, 100), profits)
plt.show()
```



There is a range for salary increases where profits are maximized. Beyond this range, the cost of salary increases outweighs the savings from reduced attrition. Excessively high raises lead to diminishing returns in profits, as increased costs are not sufficiently offset by reduced attrition. Following the suggestion of the graph, ie to reduce all wages would lead to shorter gains but will definately be harmfull for the long term success of the company. A balanced approach to salary adjustments is crucial. Both salary reductions and excessive increases can decrease profits.

By taking into account the findings, a possible action for the company would be to make moderate salary increases targeted at roles with high turnover rates or critical to the business. Moreover, it should consider non-monetary benefits and rewards to enhance employee satisfaction without significantly increasing costs.