HR ATTRIBUTION

In [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
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
```

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

In [2]:
```
df = pd.read_csv("/Users/OscarIroh_1/Downloads/CLASSWORKWEEK4/HR_Analy
```

In [3]:
```
df.head()
```

Out[3]:

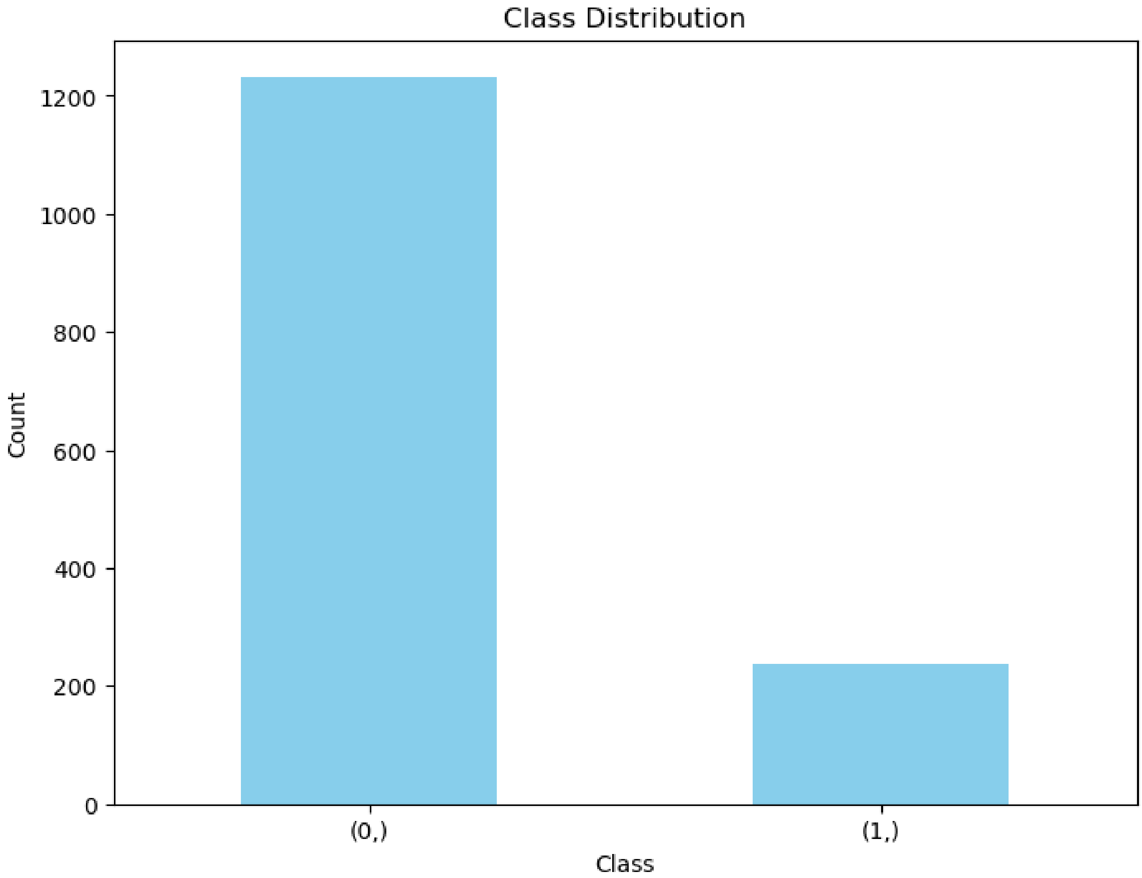
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Edu
0	41	Yes	Travel_Rarely	1102	Sales	1	2	L
1	49	No	Travel_Frequently	279	Research & Development	8	1	L
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	L
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

In [4]:
```
y = df[["Attrition"]].copy()
X = df.drop("Attrition", axis = 1)
```

In [5]:
```
y["Attrition"] = [1 if i == "Yes" else 0 for i in y["Attrition"]]
```

In [6]:
```
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()
```



In [7]:
```
# 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,dr
```

In [8]:
```
x_train,x_test,y_train,y_test=train_test_split(X,
 y, test_size=0.20, random_state=42)
```

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

In [9]:
```
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 :  0.79
```

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

In [10]:
```

# 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_g

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)

```

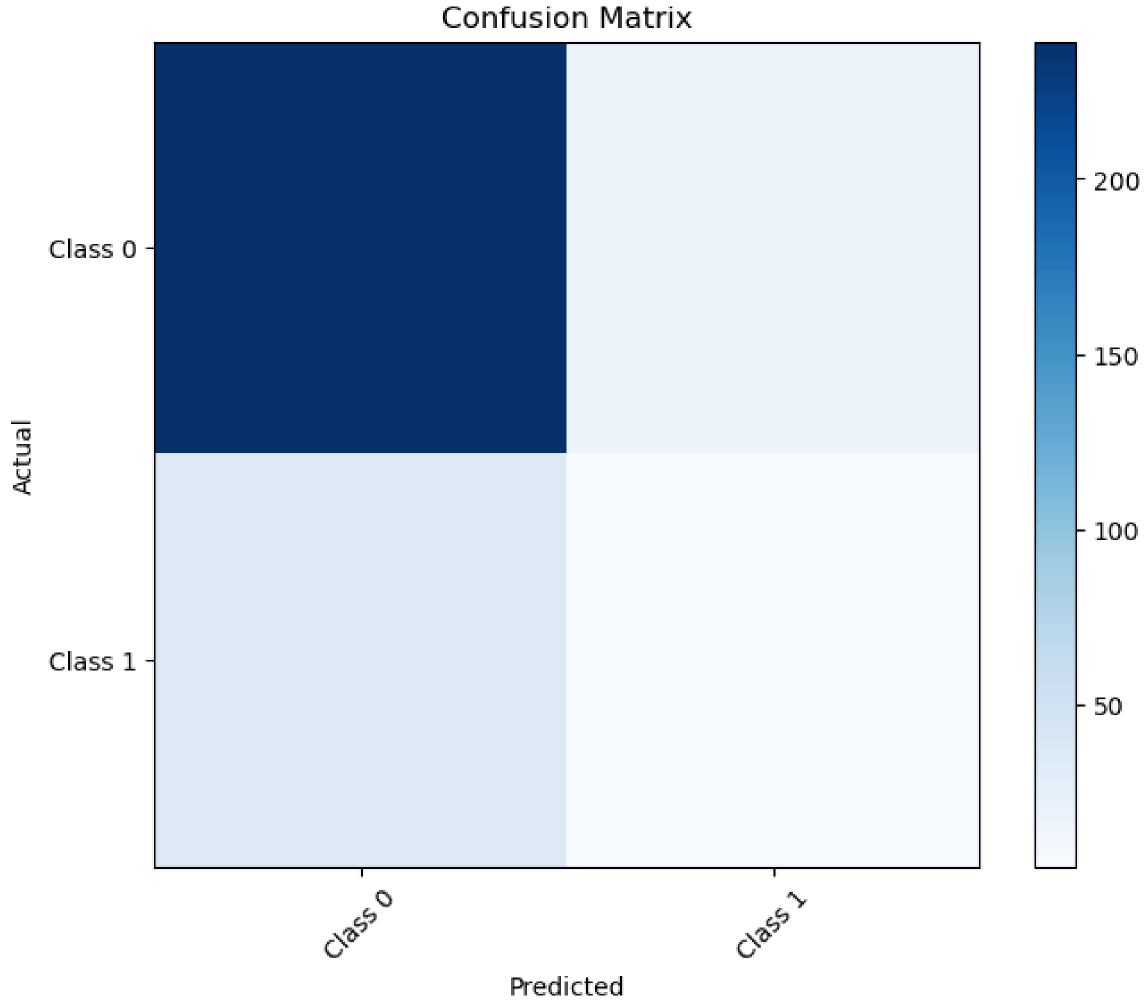
```
Best Parameters: {'criterion': 'gini', 'max_depth': 6, 'min_samples_
leaf': 2, 'min_samples_split': 2}
Best F1-Score: 0.8214764475510983
```

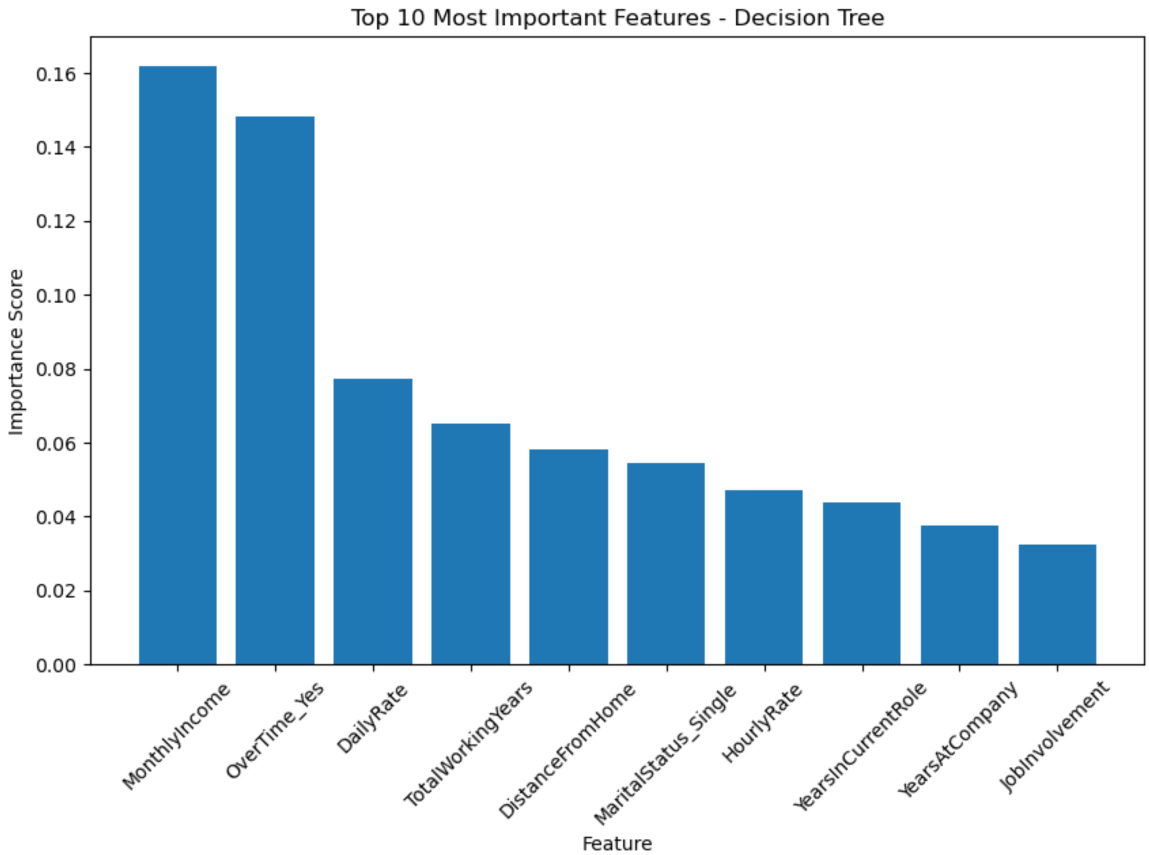
In [11]:
```
clf = tree.DecisionTreeClassifier(**best_params, random_state =42)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_train)
acc=accuracy_score(y_train,y_pred)
print("IN SAMPLE ACCURACY : " , round(acc,2))

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

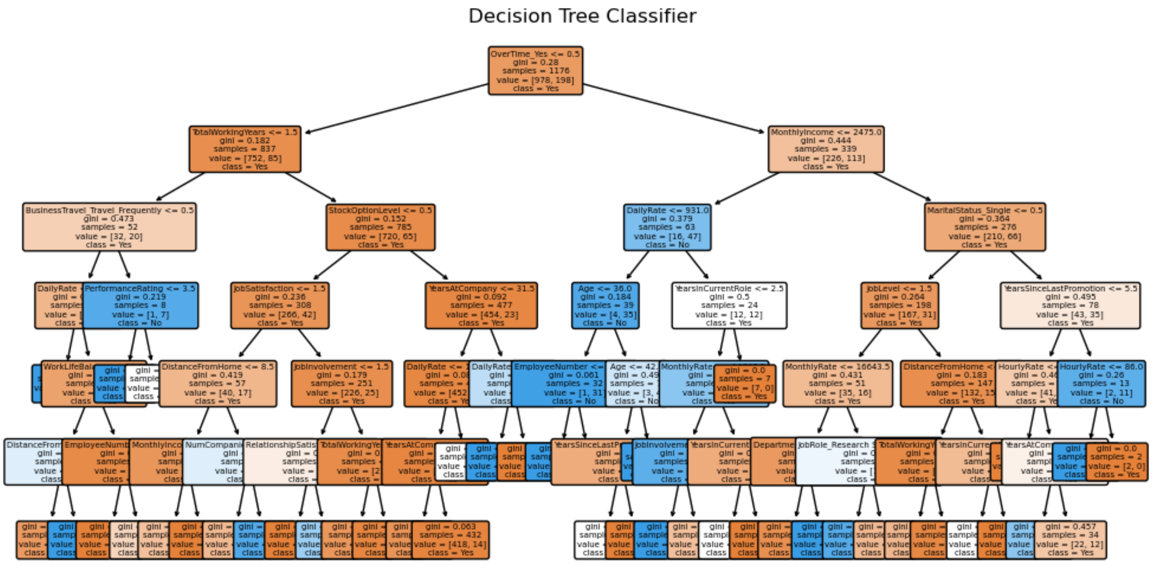
```
IN SAMPLE ACCURACY :  0.91
OUT OF SAMPLE ACCURACY :  0.83
```

4.) Plot ......

In [12]:
```
# 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))
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 feat
plt.figure(figsize=(12, 6))
plot_tree(clf, filled = True, feature_names=X.columns.tolist(), class_
plt.title('Decision Tree Classifier')
plt.show()
```


Confusion Matrix
Class 0
Class 1
Actual
Class 0
Class 1
Predicted
200
150
100
50


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



In [ ]:

In [ ]:

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. Calculate anything you think would assist in your assessment.

ANSWER :

In [ ]:

In [13]:
```
np.corrcoef(np.array(X["OverTime_Yes"]), y["Attrition"])
```

Out[13]:
```
array([[1.        , 0.24611799],
       [0.24611799, 1.        ]])
```

In [ ]:

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

In [14]:
```
x_train_experiment = x_train.copy()
```

In [15]:
```
x_train_experiment ['OverTime_Yes'] = 0
```

In [16]:
```
y_pred_experiment = clf.predict(x_train_experiment)
y_pred = clf.predict(x_train)
```

In [17]:
```
print("Stopping overtime work would have prevented people from leaving
```

```
Stopping overtime work would have prevented people from leaving 59
```

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

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

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

What do you suggest to maximize company profits?

In [18]:
```
x_train_experiment["Y"] = y_pred
x_train_experiment["Y_exp"] = y_pred_experiment
x_train_experiment["Ret_Change"] = x_train_experiment['Y'] - x_train_e
```

In [20]:
```
# Saving: Change in Training Cost
sav = sum(x_train_experiment["Ret_Change"] * 2.8 * x_train_experiment[
```

In [22]:
```
cost = 2000 * len(x_train[x_train['OverTime_Yes'] == 1])
```

In [23]:
```
print("Profit from this experiment: ", sav-cost)
```

```
Profit from this experiment:  -117593.99999999977
```

ANSWER : It will lead to a loss in money to let nobody work overtime, therefore we should continue giving overtime

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

In [27]:

```
 raise_amount = 500

x_train_experiment = x_train.copy()
x_train_experiment["MonthlyIncome"] = x_train_experiment['MonthlyIncom

# Make predictions
y_pred_experiment = clf.predict(x_train_experiment)
y_pred = clf.predict(x_train)

# Calculate Retention Change
x_train_experiment["Retention_Change"] = y_pred - y_pred_experiment

# Print Retention Difference
print("Retention difference: ", sum(x_train_experiment['Retention_Char

# Calculate and print profit from the experiment
profit = sum(x_train_experiment["Retention_Change"] * 2.8 * x_train_ex
cost = raise_amount * len(x_train)

print("Profit from this experiment: ", profit - cost)
```

```
Retention difference:  22
Profit from this experiment:  -416449.6000000001
```

We retained 22 employees but profits were lost so it wasnt a worth the increase

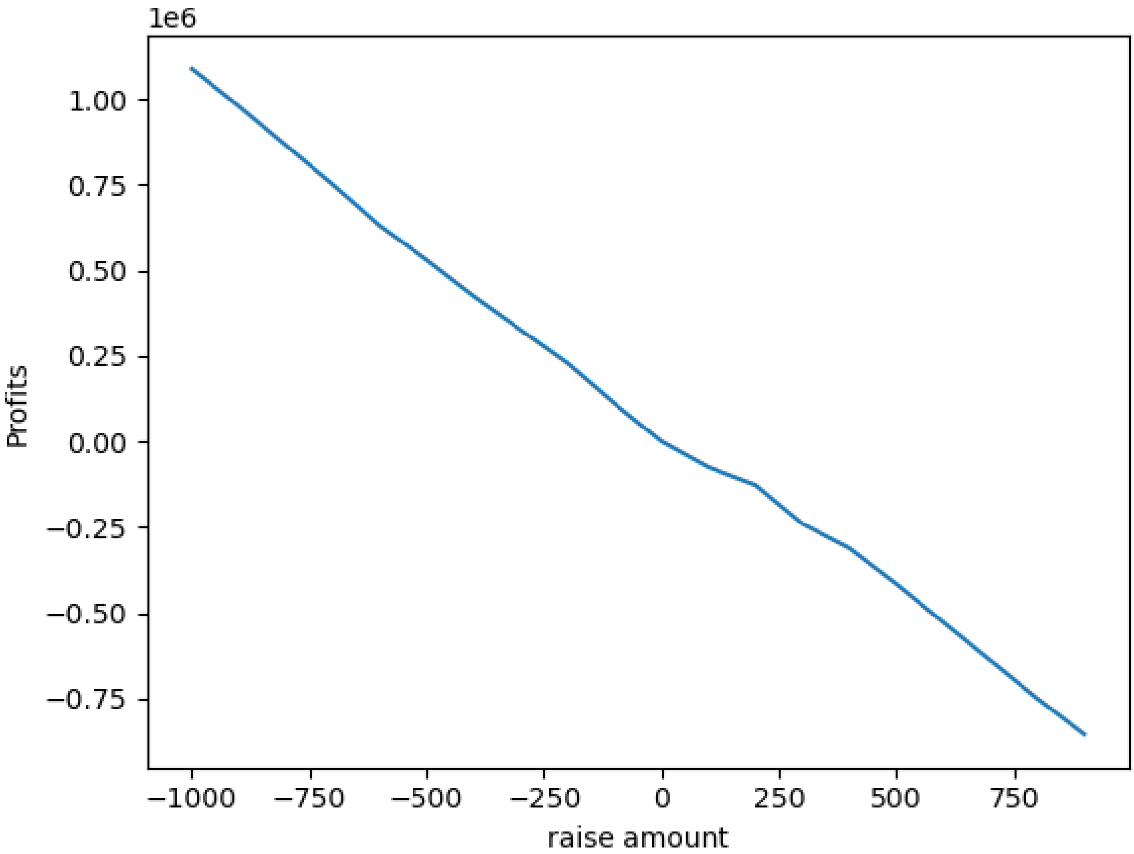
In [31]:
```
profits = []
raise_amounts = []
for raise_amount in range(-1000, 1000, 100):
    x_train_experiment = x_train.copy()
    x_train_experiment["MonthlyIncome"] = x_train_experiment['MonthlyI
    y_pred_experiment = clf.predict(x_train_experiment)
    y_pred = clf.predict(x_train)
    x_train_experiment["Y"] = y_pred
    x_train_experiment["Y_exp"] = y_pred_experiment
    x_train_experiment["Retention_change"] = x_train_experiment['Y'] -

    # Savings: Change in Training Costs
    print("Retention difference: ", sum(x_train_experiment['Retention_

    sav = sum(x_train_experiment["Retention_change"] * 2.8 * x_train_e
    cost = raise_amount * len(x_train)

    print("Profit from this experiment: ", sav - cost)
    profits.append(sav - cost)
    raise_amounts.append(raise_amount)
```

```
Retention difference:  -16
Profit from this experiment:  1087584.4
Retention difference:  -14
Profit from this experiment:  979524.0
Retention difference:  -13
Profit from this experiment:  864992.8
Retention difference:  -12
Profit from this experiment:  750738.8
Retention difference:  -12
Profit from this experiment:  629778.8
Retention difference:  -9
Profit from this experiment:  530138.0
Retention difference:  -7
Profit from this experiment:  424200.0
Retention difference:  -4
Profit from this experiment:  326096.4
Retention difference:  -1
Profit from this experiment:  228440.8
Retention difference:  -1
Profit from this experiment:  110714.8
Retention difference:  0
Profit from this experiment:  0.0
Retention difference:  6
Profit from this experiment:  -75328.40000000001
Retention difference:  15
Profit from this experiment:  -127503.60000000002
Retention difference:  15
Profit from this experiment:  -240914.8
Retention difference:  21
Profit from this experiment:  -311586.80000000005
Retention difference:  22
Profit from this experiment:  -416449.6000000001
Retention difference:  22
Profit from this experiment:  -527889.6000000001
Retention difference:  22
Profit from this experiment:  -639329.6000000001
Retention difference:  22
Profit from this experiment:  -750769.6000000001
Retention difference:  23
Profit from this experiment:  -854999.6000000001
```

ANSWER :

In [32]:
```
plt.xlabel("raise amount")
plt.ylabel("Profits")
plt.plot(raise_amounts, profits)
```

Out[32]: [<matplotlib.lines.Line2D at 0x15e503650>]



The graph above illustrates a negative relationship, indicating that increasing the monthly income consistently results in decreased profits, even when considering the cost associated with lower retention

In [ ]: