project

April 26, 2024

1 Evaluation code for filter_dp

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1.0.1 Load and clean datasets

[2]:
```
# Load input datasets
input_salaries = pd.read_csv('input/EmployeeSalaries.csv',␣
↪names=["Department","Department_Name","Division","Gender","Base_Salary","Overtime_Pay","Lon
input_students = pd.read_csv('input/StudentsPerformance.csv',␣
↪names=["gender","race_ethnicity","parental_education","lunch","test_preparation","math_scor

# Load output datasets
output_salaries = pd.read_csv('output/EmployeeSalaries.perturbed.csv',␣
↪names=["time","Department","Department_Name","Division","Gender","Base_Salary","Overtime_Pa
output_students = pd.read_csv('output/StudentsPerformance.perturbed.csv',␣
↪names=["time","gender","race_ethnicity","parental_education","lunch","test_preparation","ma

# Remove the last line from the "student" input/output files as that's a␣
↪control line
input_students.drop(input_students.tail(1).index,inplace=True)
output_students.drop(output_students.tail(1).index,inplace=True)
```

1.0.2 Sensitivty Calculations (used in creation of sample settings)

[3]:
```
# The lower the c, the less outliers (notable out-of-range values) will be used
def calculate_sensitivity(series, c=0.25):
    return c * series.std()

math_sensitivity = calculate_sensitivity(input_students['math_score'])
reading_sensitivity = calculate_sensitivity(input_students['reading_score'])
writing_sensitivity = calculate_sensitivity(input_students['writing_score'])
print(f'Sensitivity of math_score: {math_sensitivity}')
print(f'Sensitivity of reading_score: {reading_sensitivity}')
```

```
print(f'Sensitivity of writing_score: {writing_sensitivity}')
```

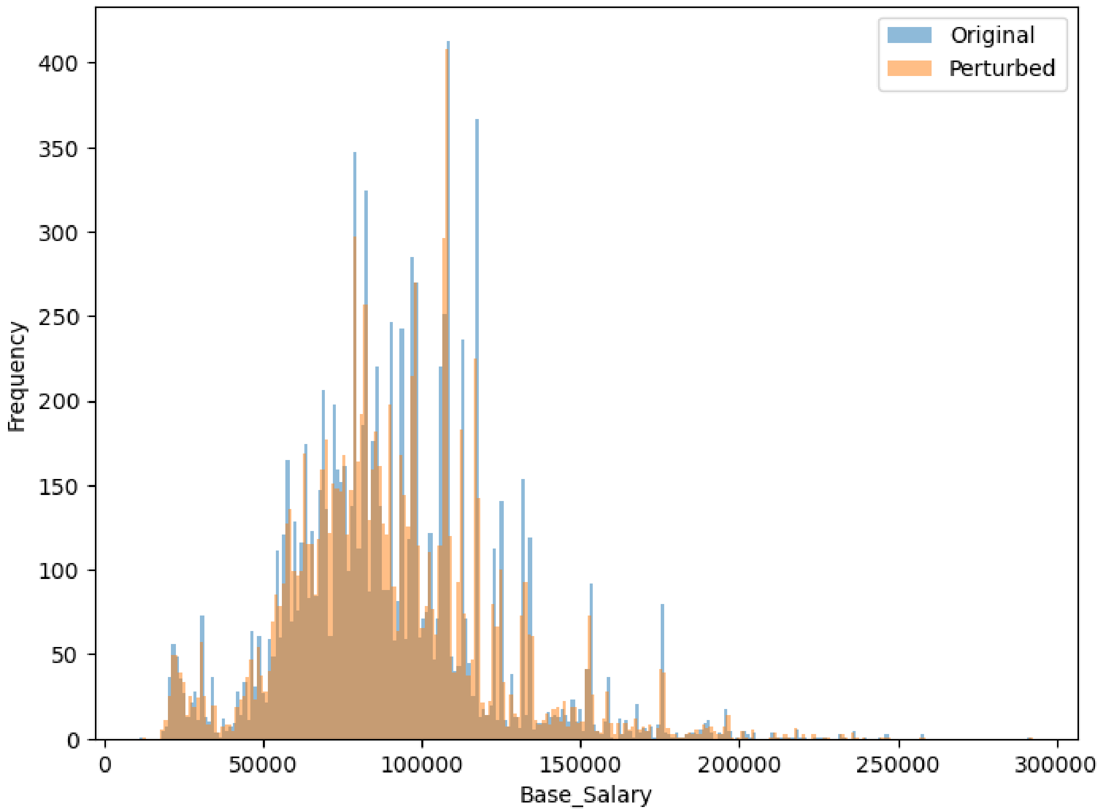
```
Sensitivity of math_score: 3.790770024002367
Sensitivity of reading_score: 3.650047984313055
Sensitivity of writing_score: 3.7989142527174105
```

[4]:

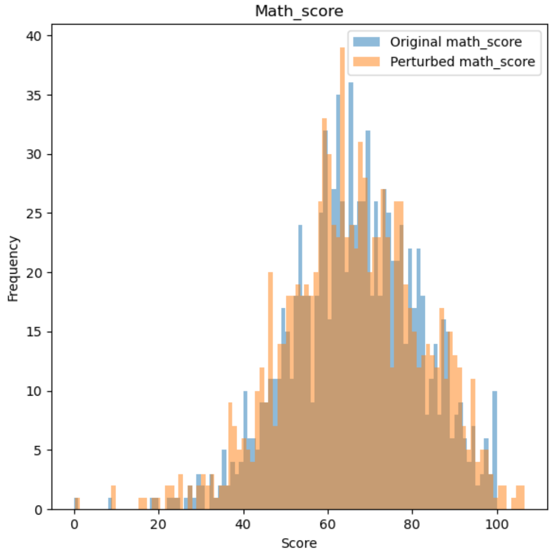
```
# Salaries dataset analysis
print("Salaries Dataset:")
print("Original Base_Salary - Mean: {:.2f}, Std: {:.2f}".
 ↪format(input_salaries['Base_Salary'].mean(), input_salaries['Base_Salary'].
 ↪std()))
print("Perturbed Base_Salary - Mean: {:.2f}, Std: {:.2f}".
 ↪format(output_salaries['Base_Salary'].mean(), output_salaries['Base_Salary'].
 ↪std()))

plt.figure(figsize=(8, 6))
plt.hist(input_salaries['Base_Salary'], bins=250, alpha=0.5, label='Original')
plt.hist(output_salaries['Base_Salary'], bins=250, alpha=0.5, label='Perturbed')
plt.xlabel('Base_Salary')
plt.ylabel('Frequency')
plt.legend()
plt.show()

# Students dataset analysis
print("Students Dataset:")
fig, axs = plt.subplots(2, 2, figsize=(12, 12))

# Loop through each score to plot and print statistics
scores = ['math_score', 'reading_score', 'writing_score']
data = []

for index, score in enumerate(scores):
    row, col = index // 2, index % 2
    # Calculate statistics
    original_mean = input_students[score].mean()
    original_std = input_students[score].std()
    perturbed_mean = output_students[score].mean()
    perturbed_std = output_students[score].std()

    # Prepare statistics for printing on the plot and in console
    print("Original {} - Mean: {:.2f}, Std: {:.2f}".format(score,␣
 ↪original_mean, original_std))
    print("Perturbed {} - Mean: {:.2f}, Std: {:.2f}".format(score,␣
 ↪perturbed_mean, perturbed_std))

    # Adding data for the table
```

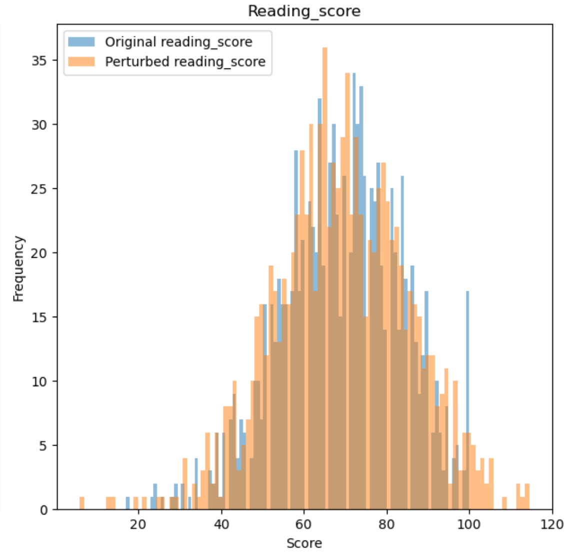
```
    data.append([score, f"{original_mean:.2f}", f"{original_std:.2f}",␣
↪f"{perturbed_mean:.2f}", f"{perturbed_std:.2f}"])

    # Plotting the histogram for each score
    axs[row, col].hist(input_students[score], bins=100, alpha=0.5,␣
↪label='Original ' + score)
    axs[row, col].hist(output_students[score], bins=100, alpha=0.5,␣
↪label='Perturbed ' + score)
    axs[row, col].set_title(score.capitalize())
    axs[row, col].set_xlabel('Score')
    axs[row, col].set_ylabel('Frequency')
    axs[row, col].legend()

# Use the last subplot to create a table
axs[1, 1].axis('on')  # Turn on the axis
axs[1, 1].axis('off')  # Turn off the axis lines and labels

# Creating a table
column_labels = ["Score", "Original\nMean", "Original\nStd", "Perturbed\nMean",␣
↪"Perturbed\nStd"]
table = axs[1, 1].table(cellText=data, colLabels=column_labels, loc='center',␣
↪cellLoc='center')
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1, 2)  # Scale table size to fit better

plt.tight_layout()
plt.show()
```

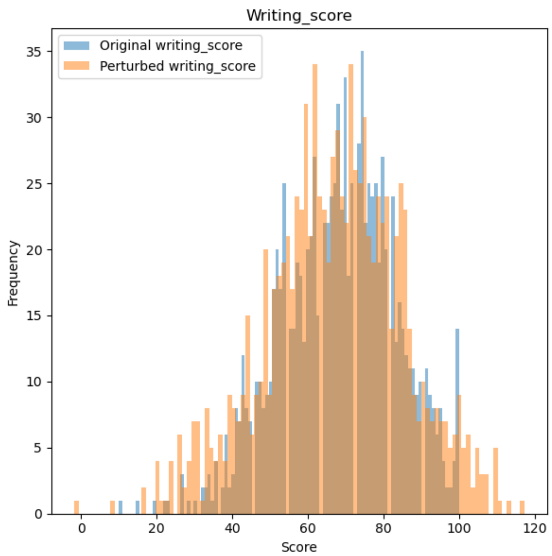
```
Salaries Dataset:
Original Base_Salary - Mean: 90312.17, Std: 31240.84
Perturbed Base_Salary - Mean: 90314.54, Std: 31246.67
```




```
Students Dataset:
Original math_score - Mean: 66.09, Std: 15.16
Perturbed math_score - Mean: 66.09, Std: 16.20
Original reading_score - Mean: 69.17, Std: 14.60
Perturbed reading_score - Mean: 69.23, Std: 16.05
Original writing_score - Mean: 68.05, Std: 15.20
Perturbed writing_score - Mean: 67.11, Std: 18.30
```







Score	Original Mean	Original Std	Perturbed Mean	Perturbed Std
math_score	66.09	15.16	66.09	16.20
reading_score	69.17	14.60	69.23	16.05
writing_score	68.05	15.20	67.11	18.30

[5]:
```python
# Calculate the mean absolute error (MAE) for each perturbed attribute
mae_base_salary = np.mean(np.abs(input_salaries['Base_Salary'] - output_salaries['Base_Salary']))
mae_math_score = np.mean(np.abs(input_students['math_score'] - output_students['math_score']))
mae_reading_score = np.mean(np.abs(input_students['reading_score'] - output_students['reading_score']))
mae_writing_score = np.mean(np.abs(input_students['writing_score'] - output_students['writing_score']))

print(f"Mean Absolute Error (MAE) for Base_Salary: {mae_base_salary:.2f}")
print(f"Mean Absolute Error (MAE) for math_score: {mae_math_score:.2f}")
```

```python
print(f"Mean Absolute Error (MAE) for reading_score: {mae_reading_score:.2f}")
print(f"Mean Absolute Error (MAE) for writing_score: {mae_writing_score:.2f}")

# Calculate the mean squared error (MSE) for each perturbed attribute
mse_base_salary = np.mean((input_salaries['Base_Salary'] - output_salaries['Base_Salary'])**2)
mse_math_score = np.mean((input_students['math_score'] - output_students['math_score'])**2)
mse_reading_score = np.mean((input_students['reading_score'] - output_students['reading_score'])**2)
mse_writing_score = np.mean((input_students['writing_score'] - output_students['writing_score'])**2)

print(f"\nMean Squared Error (MSE) for Base_Salary: {mse_base_salary:.2f}")
print(f"Mean Squared Error (MSE) for math_score: {mse_math_score:.2f}")
print(f"Mean Squared Error (MSE) for reading_score: {mse_reading_score:.2f}")
print(f"Mean Squared Error (MSE) for writing_score: {mse_writing_score:.2f}")

# Calculate the root mean squared error (RMSE) for each perturbed attribute
rmse_base_salary = np.sqrt(mse_base_salary)
rmse_math_score = np.sqrt(mse_math_score)
rmse_reading_score = np.sqrt(mse_reading_score)
rmse_writing_score = np.sqrt(mse_writing_score)

print(f"\nRoot Mean Squared Error (RMSE) for Base_Salary: {rmse_base_salary:.2f}")
print(f"Root Mean Squared Error (RMSE) for math_score: {rmse_math_score:.2f}")
print(f"Root Mean Squared Error (RMSE) for reading_score: {rmse_reading_score:.2f}")
print(f"Root Mean Squared Error (RMSE) for writing_score: {rmse_writing_score:.2f}")

# Plot the distribution of errors for each perturbed attribute
fig, axs = plt.subplots(2, 2, figsize=(12, 8))
axs[0, 0].hist(input_salaries['Base_Salary'] - output_salaries['Base_Salary'], bins=250)
axs[0, 0].set_title('Error Distribution for Base_Salary')
axs[0, 1].hist(input_students['math_score'] - output_students['math_score'], bins=50)
axs[0, 1].set_title('Error Distribution for math_score')
axs[1, 0].hist(input_students['reading_score'] - output_students['reading_score'], bins=50)
axs[1, 0].set_title('Error Distribution for reading_score')
axs[1, 1].hist(input_students['writing_score'] - output_students['writing_score'], bins=50)
axs[1, 1].set_title('Error Distribution for writing_score')
```

```
plt.tight_layout()
plt.show()
```

```
Mean Absolute Error (MAE) for Base_Salary: 421.58
Mean Absolute Error (MAE) for math_score: 3.87
Mean Absolute Error (MAE) for reading_score: 4.88
Mean Absolute Error (MAE) for writing_score: 7.52

Mean Squared Error (MSE) for Base_Salary: 280631.57
Mean Squared Error (MSE) for math_score: 29.93
Mean Squared Error (MSE) for reading_score: 47.46
Mean Squared Error (MSE) for writing_score: 108.12

Root Mean Squared Error (RMSE) for Base_Salary: 529.75
Root Mean Squared Error (RMSE) for math_score: 5.47
Root Mean Squared Error (RMSE) for reading_score: 6.89
Root Mean Squared Error (RMSE) for writing_score: 10.40
```