Using Differential Privacy on Extended Dataset

Differential Privacy Code

- **Loads the dataset** (fintechHackathonDataset.csv).
- **Splits and normalizes** the dataset.
- **Adds Laplace noise** to create multiple differentially private datasets.
- **Trains a RandomForestClassifier** on:
  - ✓The original dataset
  - ✓The noisy datasets with varying privacy levels (ε).
- **Evaluates the impact of noise** on classification accuracy.

Function: read_in_files

```
def read_in_files():
    data = pd.read_csv(file_path)
    return data
```

- Reads the dataset from fintechHackathonDataset.csv
  - ✓Reads a CSV file into a Pandas DataFrame.
  - ✓Returns the DataFrame to be used later in the script.

Function: Train_test_split_with_normalization

```python
def Train_test_split_with_normalization(data):
    target_column = "DEFAULT"
    X = data.drop(columns=[target_column])
    y = data[target_column]

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state

    # Normalize the training data
    scaler = StandardScaler()
    X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train), index=X_train.index, colu
    X_test_scaled = pd.DataFrame(scaler.transform(X_test), index=X_test.index, columns=X_t

    # Define a helper function for adding Laplace noise
    def add_noise(data, epsilon, sensitivity=1.0):
        scale = sensitivity / epsilon
        noise = np.random.laplace(0, scale, data.shape)
        return data + noise
```

- Splits the dataset into training and test sets.
- Normalizes the features using StandardScaler.
- Adds Laplace noise to simulate differential privacy.

- Reduces ε iteratively, each time halving its value.
- Generates multiple noisy datasets for model training.

```python
epsilon = epsilonInitialValue
print("The value of epsilon is:", epsilon)
X_train_noisy_epsilon = add_noise(X_train_scaled, epsilon)

epsilon = epsilon / 2
print("The value of epsilon is:", epsilon)
X_train_noisy_epsilonOver2 = add_noise(X_train_scaled, epsilon)

...

epsilon = epsilon / 2
print("The value of epsilon is:", epsilon)
X_train_noisy_epsilonOver512 = add_noise(X_train_scaled, epsilon)
```

encrypt

Why might normalization be needed?

- Ensures Fair Comparisons Between Features
  - ✓ Features in datasets often have different ranges (e.g., income in thousands vs. age in years).
  - ✓ Without normalization, larger magnitude features dominate, leading to biased models.
- Improves Machine Learning Model Performance
  - ✓ Many models perform better when features are on the same scale.
  - ✓ It prevents some features from overpowering others due to larger numerical values.
- Enhances Gradient Descent Convergence
  - ✓ Optimization algorithms like gradient descent work best when data is scaled.
  - ✓ If features have different scales, gradient updates become uneven, leading to slow or unstable convergence.
- Essential for Distance-Based Algorithms
  - ✓ Algorithms like k-NN, k-Means and PCA rely on distance metrics (e.g., Euclidean distance).
  - ✓ If data isn't normalized, features with larger scales dominate distance calculations, making clustering or classification ineffective.
- Helps with Differential Privacy Mechanisms
  - ✓ The function later adds Laplace noise to protect privacy
  - ✓ If data isn't normalized, noise affects features unevenly, reducing privacy guarantees and model utility.

Function: runRandomForestClassifierModels

```
def runRandomForestClassifierModels(...):
    RandomForestClassifierRunForResults(X_train, y_train,  X_test, y_test)
    RandomForestClassifierRunForResults(X_train_noisy_epsilon50, y_train,  X_test, y_test)
    ...
    RandomForestClassifierRunForResults(X_train_noisy_epsilon05, y_train,  X_test, y_test)
```

- Runs the RandomForestClassifier on multiple noisy versions of the dataset.
  - ✓Calls RandomForestClassifierRunForResults, which trains and evaluates a standard Random Forest model without noise.
  - ✓Iterates through training data with decreasing epsilon values
    - Smaller epsilon = More noise = Stronger privacy but lower accuracy.
- It can be used to compares the impact of differential privacy noise on model accuracy.

Functions: RandomForestClassifierRunForResults and fit_modelRandomForestClassifier

```python
def RandomForestClassifierRunForResults(X_train, y_train, X_test, y_test):
    total_plain = 0.0
    numberOfIterations = 0

    for i in range(max_iterations):
        model = fit_modelRandomForestClassifier(X_train, y_train)
        a = return_accuracy(model, X_test, y_test)
        total_plain += a
        numberOfIterations += 1

    avg_plain = total_plain / numberOfIterations
    print(f"The average accuracy of the plain RandomForestClassifier model is: {avg_plain:.4
```

- This function trains and evaluates a RandomForest model max_iterations times.
- It calculates the average accuracy.

- This function trains a RandomForestClassifier with 100 trees.
- Where later its accuracy is found

```python
def fit_modelRandomForestClassifier(X_train, y_train):
    rf = RandomForestClassifier(n_estimators=100).fit(X_train, y_train)
    return rf
```

```python
def return_accuracy(gnb, X_test, y_test):
    y_pred = gnb.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy
```

encrypt

Function: fit_modelRandomForestClassifier_Hyperparameters

```
def fit_modelRandomForestClassifier_Hyperparameters(X_train, y_train):
    param_grid = {
        'n_estimators': [10, 50, 100, 150, 200, 250],
        'max_depth': [10, 5, 3, None],
        'min_samples_split': [2, 5, 7, 10],
        'min_samples_leaf': [1, 4, 6, 9],
        'max_features': ['sqrt', 'log2', 0.5],
        'bootstrap': [True]
    }
    rf = RandomForestClassifier()
    grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, n_jobs=-1)
    grid_search.fit(X_train, y_train)

    return grid_search.best_estimator_
```

- This function is not called in the code
  - ✓ If you want, you can try, alter the code for this function to also/instead run
  - ✓ Keep the max_iterations value low as otherwise it will take time

- This function tunes hyperparameters of RandomForestClassifier using GridSearchCV.
- It selects the best model based on cross-validation.

Other coding exercises you can try

- GaussianNBModels and DecisionTreeClassifiers
- Models you can try yourself
  - ✓Function call is commented out in the main function
  - ✓Code for this is similar to what has been presented
  - ✓Appropriate functions will need to instead be called
- You can try these out with/without hyperparameters
- You can identify the most accurate model, and serialise it as a .pkl file
- Try these for yourself – and contact me for any help.

IBM Diffprivlib

- Current approach adds Laplace noise directly to the normalized feature values before training the model
- IBM's **Diffprivlib** integrates DP directly into machine learning models
  - ✓Instead of adding noise to the raw dataset, Diffprivlib modifies the learning algorithm itself
  - ✓Models use differential privacy internally, adding noise to model parameters instead of modifying data.
- **Diffprivlib** solutions may be better to use in some applications
  - ✓And easier to understand and use programmatically

- We presented another solution which is more relative to Local Differential Privacy – addition of noise to dataset, and then analytics which can be carried out by an external party

Thank you!

https://encrypt-project.eu/

encrypt-project

@encrypt_project

encrypt