Task 2: Logistic Regression Model Report

1. Task Description

This task involved implementing a logistic regression model to predict binary outcomes (customer churn). The objectives were to load and preprocess the dataset, train a logistic regression model using scikit-learn, interpret model coefficients and odds ratios, and evaluate the model using metrics such as accuracy, precision, recall, and the ROC curve. Subsequently, the model was improved by addressing class imbalance using SMOTE and optimizing hyperparameters with GridSearchCV.

2. Datasets Used

The datasets used for this task were:

- `churn-bigml-80.csv` (for training the model)
- `churn-bigml-20.csv` (for testing the model)

These files likely represent an 80/20 split of a larger customer churn dataset.

3. Model Implemented

A Logistic Regression model from `scikit-learn` was implemented.

4. Features and Target

- **Target Variable (y):** `Churn` (binary: 1 for churn, 0 for no churn)
- **Features (X):** All other columns in the dataset, including `State`, `Account length`, `Area code`, `International plan`, `Voice mail plan`, `Number vmail messages`, `Total day minutes`, `Total day calls`, `Total day charge`, `Total eve minutes`, `Total eve calls`, `Total eve charge`, `Total night minutes`, `Total night calls`, `Total night charge`, `Total intl minutes`, `Total intl calls`, `Total intl charge`, and `Customer service calls`.

5. Preprocessing Steps

1. **Loading Data:** Training and testing datasets were loaded using pandas `read_csv`.
2. **Categorical Feature Conversion:**
   - `International plan` and `Voice mail plan` were converted from 'Yes'/'No' to 1/0.
   - `Churn` (target variable) was converted from True/False to 1/0.
3. **One-Hot Encoding:** The `State` feature was one-hot encoded using `pd.get_dummies` to convert categorical state names into numerical format, with `drop_first=True` to avoid multicollinearity.
4. **Column Alignment:** Columns in the training and testing datasets were aligned to ensure consistency after one-hot encoding.
5. **Class Imbalance Handling (Optimization Phase):** SMOTE (Synthetic Minority Over-sampling Technique) was applied to the training data to balance the number of samples in the minority class (`Churn=1`) and the majority class (`Churn=0`).

6. Model Training and Evaluation

Initial Model (without SMOTE and GridSearchCV)

- **Training Data Size:** 2666 samples
- **Testing Data Size:** 667 samples

Evaluation Metrics:

- **Accuracy:** 0.8486
- **Precision:** 0.4118
- **Recall:** 0.1474
- **ROC AUC:** 0.7875

Confusion Matrix:

```
[[552  20]
 [ 81  14]]
```

Optimized Model (with SMOTE and GridSearchCV)

- **Resampled Training Data Size:** 4556 samples (balanced classes)
- **Testing Data Size:** 667 samples (original, unbalanced)

Best Hyperparameters Found:

- `C` : 0.1
- `max_iter` : 1000
- `solver` : `liblinear`

Evaluation Metrics:

- **Accuracy:** 0.7766
- **Precision:** 0.3247
- **Recall:** 0.5263
- **F1-Score:** 0.4016
- **ROC AUC:** 0.7649

Confusion Matrix:

```
[[468 104]
 [ 45  50]]
```

7. Interpretation of Results

The initial logistic regression model showed high accuracy (0.8486) but very low recall (0.1474). This indicated that while the model was good at correctly identifying non-churners, it struggled significantly to identify actual churners, leading to a high number of false negatives (81). The ROC AUC of 0.7875 suggested a reasonable ability to distinguish between classes overall.

To address the low recall, SMOTE was applied to balance the training dataset, and GridSearchCV was used to find optimal hyperparameters for the logistic regression model. The optimized model demonstrated a significant improvement in **recall**, increasing from 0.1474 to **0.5263**. This means the model is now much more effective at identifying customers who are likely to churn.

However, this improvement came with a trade-off:

- **Precision** decreased from 0.4118 to 0.3247, indicating an increase in false positives (customers predicted to churn who do not). This is evident in the confusion matrix, where false positives increased from 20 to 104.
- **Overall Accuracy** also decreased from 0.8486 to 0.7766.
- The **ROC AUC** slightly decreased from 0.7875 to 0.7649.

The **F1-Score** of 0.4016 for the optimized model (which was not explicitly calculated for the initial model but would have been lower due to very low recall) suggests a better balance between precision and recall, which is often a more appropriate metric for imbalanced classification problems like churn

prediction. Depending on the business objective (e.g., minimizing missed churners vs. minimizing false alarms), the optimized model with higher recall might be preferred.

8. Script Execution and Output

Below is the console output from the execution of `logistic_regression_model.py` after the modifications for optimization.

```
Training dataset loaded successfully. First 5 rows:
  State  Account length  ...  Customer service calls  Churn
0    KS             128  ...                       1  False
1    OH             107  ...                       1  False
2    NJ             137  ...                       0  False
3    OH              84  ...                       2  False
4    OK              75  ...                       3  False

[5 rows x 20 columns]

Testing dataset loaded successfully. First 5 rows:
  State  Account length  ...  Customer service calls  Churn
0    LA             117  ...                       1  False
1    IN              65  ...                       4   True
2    NY             161  ...                       4   True
3    SC             111  ...                       2  False
4    HI              49  ...                       1  False

[5 rows x 20 columns]

Training Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2666 entries, 0 to 2665
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   2666 non-null   object
 1   Account length          2666 non-null   int64
 2   Area code               2666 non-null   int64
 3   International plan      2666 non-null   object
 4   Voice mail plan         2666 non-null   object
 5   Number vmail messages   2666 non-null   int64
 6   Total day minutes       2666 non-null   float64
 7   Total day calls         2666 non-null   int64
 8   Total day charge        2666 non-null   float64
 9   Total eve minutes       2666 non-null   float64
 10  Total eve calls         2666 non-null   int64
 11  Total eve charge        2666 non-null   float64
 12  Total night minutes     2666 non-null   float64
 13  Total night calls       2666 non-null   int64
 14  Total night charge      2666 non-null   float64
 15  Total intl minutes      2666 non-null   float64
 16  Total intl calls        2666 non-null   int64
```

```
 17  Total intl charge       2666 non-null   float64
 18  Customer service calls  2666 non-null   int64
 19  Churn                   2666 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 398.5+ KB

Testing Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   667 non-null    object
 1   Account length          667 non-null    int64
 2   Area code               667 non-null    int64
 3   International plan      667 non-null    object
 4   Voice mail plan         667 non-null    object
 5   Number vmail messages   667 non-null    int64
 6   Total day minutes       667 non-null    float64
 7   Total day calls         667 non-null    int64
 8   Total day charge        667 non-null    float64
 9   Total eve minutes       667 non-null    float64
 10  Total eve calls         667 non-null    int64
 11  Total eve charge        667 non-null    float64
 12  Total night minutes     667 non-null    float64
 13  Total night calls       667 non-null    int64
 14  Total night charge      667 non-null   float64
 15  Total intl minutes      667 non-null   float64
 16  Total intl calls        667 non-null   int64
 17  Total intl charge       667 non-null   float64
 18  Customer service calls  667 non-null   int64
 19  Churn                   667 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 99.8+ KB

Training features (X_train) shape: (2666, 68)
Training target (y_train) shape: (2666,)
Testing features (X_test) shape: (667, 68)
Testing target (y_test) shape: (667,)

Applying SMOTE to balance the training data...
Resampled training data shape: (4556, 68)
Resampled training target distribution:
```

```
Churn
0    2278
1    2278
Name: count, dtype: int64

Performing GridSearchCV for hyperparameter tuning...
Best parameters found: {'C': 0.1, 'max_iter': 1000, 'solver': 'liblinear'}

Model training complete with best parameters.

Model Coefficients and Odds Ratios (from best model):
              Feature  Coefficient  Odds Ratio
60           State_TX     1.695473    5.449225
42           State_MS     1.597764    4.941971
38           State_ME     1.520830    4.576022
43           State_MT     1.465720    4.330662
48           State_NJ     1.412279    4.105300
..                ...          ...         ...
1           Area code    -0.004339    0.995670
12  Total night calls    -0.004388    0.995621
10   Total eve charge    -0.004453    0.995557
15   Total intl calls    -0.176040    0.838585
3     Voice mail plan    -2.369322    0.093544

[68 rows x 3 columns]
Intercept: -1.2658

Accuracy: 0.7766
Precision: 0.3247
Recall: 0.5263
F1-Score: 0.4016

Confusion Matrix:
[[468 104]
 [ 45  50]]
ROC AUC: 0.7649

ROC curve saved as 'roc_curve_optimized.png'
```