Data Processing

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load the dataset
data = pd.read_csv('customer_churn_large_dataset.csv')

# Explore the data
data.head()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   CustomerID                  100000 non-null  int64
 1   Name                        100000 non-null  object
 2   Age                         100000 non-null  int64
 3   Gender                      100000 non-null  object
 4   Location                    100000 non-null  object
 5   Subscription_Length_Months  100000 non-null  int64
 6   Monthly_Bill                100000 non-null  float64
 7   Total_Usage_GB              100000 non-null  int64
 8   Churn                       100000 non-null  int64
dtypes: float64(1), int64(5), object(3)
memory usage: 6.9+ MB
```

```
data.drop(columns=['Name'], inplace=True)

data = pd.get_dummies(data,columns=['Location',
'Gender'],drop_first=True)

data
```

```
       CustomerID  Age  Subscription_Length_Months  Monthly_Bill  \
0               1   63                          17         73.36
1               2   62                           1         48.76
2               3   24                           5         85.47
3               4   36                           3         97.94
4               5   46                          19         58.14
...           ...  ...                         ...           ...
99995       99996   33                          23         55.13
99996       99997   62                          19         61.65
99997       99998   64                          17         96.11
99998       99999   51                          20         49.25
```

```
99999      100000   27                            19         76.57

       Total_Usage_GB  Churn  Location_Houston  Location_Los
Angeles  \
0                 236      0                 0                      1

1                 172      0                 0                      0

2                 460      0                 0                      1

3                 297      1                 0                      0

4                 266      0                 0                      0

...               ...    ...               ...                    ...

99995             226      1                 1                      0

99996             351      0                 0                      0

99997             251      1                 0                      0

99998             434      1                 0                      0

99999             173      1                 0                      1


       Location_Miami  Location_New York  Gender_Male
0                   0                  0            1
1                   0                  1            0
2                   0                  0            0
3                   1                  0            0
4                   1                  0            0
...               ...                ...          ...
99995               0                  0            1
99996               0                  1            0
99997               0                  0            1
99998               0                  1            0
99999               0                  0            0

[100000 rows x 11 columns]

# Split data into features (X) and target (y)
X = data.drop('Churn', axis=1)
y = data['Churn']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Feature Engineering

```
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif

# Apply feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Feature selection (you can adjust k as needed)
k_best = SelectKBest(score_func=f_classif, k=10)
X_train_selected = k_best.fit_transform(X_train_scaled, y_train)
X_test_selected = k_best.transform(X_test_scaled)
```

Model Building

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

# Initialize the model
model = LogisticRegression()

# Train the model
model.fit(X_train_selected, y_train)

# Make predictions
y_pred = model.predict(X_test_selected)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.50585
```

```
print("Precision:", precision)
```

```
Precision: 0.5025013164823592
```

```
print("Recall:", recall)
```

```
Recall: 0.38473944158854956
```

```
print("F1-score:", f1)
```

```
F1-score: 0.4358052177884341
```

Model Optimization

```
from sklearn.model_selection import GridSearchCV

# Define hyperparameters to tune
param_grid = {'C': [0.1, 1, 10], 'penalty': ['l1', 'l2']}

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid,
cv=5, scoring='f1')

# Fit GridSearchCV on training data
grid_search.fit(X_train_selected, y_train)

# Get the best parameters and model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
```

```
C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\model_selection\_validation.py:378: FitFailedWarning:

15 fits failed out of a total of 30.
The score on these train-test partitions for these parameters will be
set to nan.
If these failures are not expected, you can try to debug them by
setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------
----------
15 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\lib\
site-packages\sklearn\model_selection\_validation.py", line 686, in
_fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\lib\
site-packages\sklearn\linear_model\_logistic.py", line 1091, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\lib\
site-packages\sklearn\linear_model\_logistic.py", line 61, in
_check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got
l1 penalty.
```

```
  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\model_selection\_search.py:953: UserWarning: One or
more of the test scores are non-finite: [       nan 0.44027354
nan 0.44033085        nan 0.44030385]
  warnings.warn(

best_model

LogisticRegression(C=1)
```

Model deployment

```
def predict_churn(new_customer_data):
    # Preprocess new_customer_data similarly to training data

    # Feature scaling
    new_customer_scaled = scaler.transform(new_customer_data)

    # Feature selection
    new_customer_selected = k_best.transform(new_customer_scaled)

    # Predict churn
    churn_prediction = best_model.predict(new_customer_selected)
    return churn_prediction
```