/Users/andrejakovlev/PycharmProjects/gpt_interpriter/.venv/bin/python /Users/andrejakovlev/PycharmProjects/gpt_interpriter/main.py
Open Interpreter will require approval before running code.

Use interpreter -y to bypass this.

Press CTRL-C to exit.

Sure, I can help with that. Here's a plan:

1. Load the datasets: We will load the training, testing, and validation datasets using pandas.
2. Preprocess the data: We will preprocess the data by encoding the categorical features using one-hot encoding or label encoding. We will also handle any missing values if they exist.
3. Train the model: We will train a CatBoost model on the training dataset. CatBoost can handle categorical features directly, so we don't need to do any additional preprocessing for them.
4. Evaluate the model: We will evaluate the model on the validation dataset using the precision-recall AUC metric. We will also calculate other classification metrics for a comprehensive evaluation.
5. Test the model: We will test the model on the testing dataset and calculate the precision-recall AUC and other classification metrics.
6. Identify important factors: We will use the feature importance functionality of CatBoost to identify the most important factors that influence whether someone clicks on an ad or not.

Let's start with loading the datasets.

```
import pandas as pd

# Load the datasets
df_train = pd.read_csv('df_train.csv')
df_test = pd.read_csv('df_test.csv')
df_val = pd.read_csv('df_val.csv')

# Display the first few rows of the training dataset
df_train.head()
```

Would you like to run this code? (y/n)

y

```
import pandas as pd

# Load the datasets
df_train = pd.read_csv('df_train.csv')
df_test = pd.read_csv('df_test.csv')
df_val = pd.read_csv('df_val.csv')

# Display the first few rows of the training dataset
df_train.head()
```

```
fc_imp_chk  fc_time_chk  utmtr  ...  minute_sin minute_cos tag_click
0          0           7      3  ...    0.292372  -0.956305         0
1          0           7      3  ...    0.402747  -0.915311         0
```

```
2          0          7     4  ...   -0.134851  -0.990866        0
3          0          7     5  ...   -0.960050  -0.279829        0
4          0          7     7  ...   -0.394744   0.918791        0
[5 rows x 13 columns]
```

The datasets have been loaded successfully. The training dataset contains 13 columns, including the target variable 'tag_click' and the features ['fc_imp_chk', 'fc_time_chk', 'utmtr', 'mm_dma', 'osName', 'model', 'hardware', 'site_id', 'dow', 'is_weekend', 'minute_sin', 'minute_cos'].

Next, let's check if there are any missing values in the datasets.

```
print('Missing values in training dataset:', df_train.isnull().sum().sum())
print('Missing values in testing dataset:', df_test.isnull().sum().sum())
print('Missing values in validation dataset:', df_val.isnull().sum().sum())
```

Would you like to run this code? (y/n)

y

```
print('Missing values in training dataset:', df_train.isnull().sum().sum())
print('Missing values in testing dataset:', df_test.isnull().sum().sum())
print('Missing values in validation dataset:', df_val.isnull().sum().sum())
```

```
Missing values in training dataset: 0
Missing values in testing dataset: 0
Missing values in validation dataset: 0
```

Great! There are no missing values in the training, testing, and validation datasets.

Now, let's move on to training the CatBoost model. We will first separate the features and the target variable in the datasets. Then, we will train the model on the training dataset and evaluate it on the validation dataset using the precision-recall AUC metric.

```
from catboost import CatBoostClassifier
from sklearn.metrics import average_precision_score

# Separate features and target variable
X_train = df_train.drop('tag_click', axis=1)
y_train = df_train['tag_click']
X_val = df_val.drop('tag_click', axis=1)
y_val = df_val['tag_click']

# Identify categorical features
cat_features = ['fc_imp_chk', 'fc_time_chk', 'utmtr', 'mm_dma', 'osName',
'model', 'hardware', 'site_id', 'dow', 'is_weekend']

# Initialize CatBoostClassifier
model = CatBoostClassifier(loss_function='Logloss', eval_metric='AUC',
cat_features=cat_features, verbose=200)

# Train model
model.fit(X_train, y_train, eval_set=(X_val, y_val))
```

```
# Predict probabilities on validation set
val_preds = model.predict_proba(X_val)[:, 1]

# Calculate precision-recall AUC
pr_auc = average_precision_score(y_val, val_preds)
pr_auc
```

Would you like to run this code? (y/n)

y

```
from catboost import CatBoostClassifier
from sklearn.metrics import average_precision_score

# Separate features and target variable
X_train = df_train.drop('tag_click', axis=1)
y_train = df_train['tag_click']
X_val = df_val.drop('tag_click', axis=1)
y_val = df_val['tag_click']

# Identify categorical features
cat_features = ['fc_imp_chk', 'fc_time_chk', 'utmtr', 'mm_dma', 'osName',
'model', 'hardware', 'site_id', 'dow', 'is_weekend']

# Initialize CatBoostClassifier
model = CatBoostClassifier(loss_function='Logloss', eval_metric='AUC',
cat_features=cat_features, verbose=200)

# Train model
model.fit(X_train, y_train, eval_set=(X_val, y_val))

# Predict probabilities on validation set
val_preds = model.predict_proba(X_val)[:, 1]

# Calculate precision-recall AUC
pr_auc = average_precision_score(y_val, val_preds)
pr_auc
```

```
Learning rate set to 0.051608
0:	test: 0.6539061 best: 0.6539061 (0)	total: 71.3ms	remaining:
1m 11s
200:	test: 0.8280016 best: 0.8280016 (200)	total: 2.83s	remaining:
11.2s
400:	test: 0.8266641 best: 0.8282413 (217)	total: 5.98s	remaining:
8.93s
600:	test: 0.8261442 best: 0.8282413 (217)	total: 9.3s	remaining:
6.18s
800:	test: 0.8243852 best: 0.8282413 (217)	total: 12.7s	remaining:
3.15s
999:	test: 0.8231791 best: 0.8282413 (217)	total: 16.2s	remaining:
0us
bestTest = 0.8282413132
bestIteration = 217
Shrink model to first 218 iterations.
<catboost.core.CatBoostClassifier object at 0x16d3d1630>
0.029011705508785724
```

The CatBoost model has been trained and evaluated on the validation dataset.

The precision-recall AUC on the validation dataset is approximately 0.029.

Next, let's test the model on the testing dataset and calculate the precision-recall AUC and other classification metrics.

```
from sklearn.metrics import classification_report

# Separate features and target variable in testing dataset
X_test = df_test.drop('tag_click', axis=1)
y_test = df_test['tag_click']

# Predict probabilities on testing set
test_preds = model.predict_proba(X_test)[:, 1]

# Calculate precision-recall AUC on testing set
pr_auc_test = average_precision_score(y_test, test_preds)

# Predict classes on testing set
test_class_preds = model.predict(X_test)

# Calculate classification metrics on testing set
report = classification_report(y_test, test_class_preds)

(pr_auc_test, report)
```

Would you like to run this code? (y/n)

y

```
from sklearn.metrics import classification_report

# Separate features and target variable in testing dataset
X_test = df_test.drop('tag_click', axis=1)
y_test = df_test['tag_click']

# Predict probabilities on testing set
test_preds = model.predict_proba(X_test)[:, 1]

# Calculate precision-recall AUC on testing set
pr_auc_test = average_precision_score(y_test, test_preds)

# Predict classes on testing set
test_class_preds = model.predict(X_test)

# Calculate classification metrics on testing set
report = classification_report(y_test, test_class_preds)

(pr_auc_test, report)
```

```
(0.023839058825561307, '              precision    recall  f1-score
support\n\n           0       1.00      0.76      0.87    190271\n
1       0.01      0.79      0.02       725\n\n    accuracy
0.76    190996\n   macro avg       0.51      0.78      0.44
190996\nweighted avg       1.00      0.76      0.86    190996\n')
```

The precision-recall AUC on the testing dataset is approximately 0.024. The classification report shows the precision, recall, and F1-score for each class, as well as the overall accuracy of the model.

The model has a high precision for the '0' class (no click) but a very low precision for the '1' class (click). This is likely due to the class imbalance in the dataset. The recall for both classes is relatively high, indicating that the model is able to correctly identify a high proportion of actual positives and negatives.

Finally, let's identify the most important factors that influence whether someone clicks on an ad or not. We will use the feature importance functionality of CatBoost for this.

```
feature_importances = model.get_feature_importance(prettified=True)
feature_importances
```

Would you like to run this code? (y/n)

y

```
feature_importances = model.get_feature_importance(prettified=True)
feature_importances
```

	Feature Id	Importances
0	site_id	59.733917
1	osName	7.124474
2	mm_dma	6.050077
3	fc_imp_chk	5.541812
4	dow	4.827886
5	model	4.327048
6	utmtr	4.132369
7	minute_sin	2.879160
8	hardware	2.278468
9	minute_cos	1.695488
10	is_weekend	1.030095
11	fc_time_chk	0.379205

The most important factors that influence whether someone clicks on an ad or not, ranked from most to least important, are:

1. 'site_id': The ID of the site where the ad is displayed.
2. 'osName': The name of the operating system of the user's device.
3. 'mm_dma': A feature related to the user's location.
4. 'fc_imp_chk': A categorical feature.
5. 'dow': The day of the week.
6. 'model': The model of the user's device.
7. 'utmtr': A categorical feature.
8. 'minute_sin': A feature related to the time of the ad display.
9. 'hardware': The hardware of the user's device.
10. 'minute_cos': Another feature related to the time of the ad display.
11. 'is_weekend': Whether the ad is displayed on a weekend or not.
12. 'fc_time_chk': Another categorical feature.

These factors can be used to optimize ad displays to increase the likelihood of clicks.