```
import numpy as np
import pandas as pd
```

```
data=pd.read_csv("/content/creditcard.csv")
```

```
data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0

5 rows × 31 columns

```
data.isnull()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Cla
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5969	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
5970	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
5971	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
5972	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
5973	False	False	False	False	False	False	False	False	False	False	...	True	True	True	True	True	True	True	True	True	Tr

5974 rows × 31 columns

```
data.isnull().sum()
```

	0
Time	0
V1	0
V2	0
V3	0
V4	0
V5	0
V6	0
V7	0
V8	0
V9	0
V10	0
V11	0
V12	0
V13	0
V14	0
V15	0
V16	0
V17	0
V18	1
V19	1
V20	1
V21	1
V22	1
V23	1
V24	1
V25	1
V26	1
V27	1
V28	1
Amount	1
Class	1

dtype: int64

```
data=data.dropna()
```

```
data.isnull().sum()
```

	0
Time	0
V1	0
V2	0
V3	0
V4	0
V5	0
V6	0
V7	0
V8	0
V9	0
V10	0
V11	0
V12	0
V13	0
V14	0
V15	0
V16	0
V17	0
V18	0
V19	0
V20	0
V21	0
V22	0
V23	0
V24	0
V25	0
V26	0
V27	0
V28	0
Amount	0
Class	0

dtype: int64

```
data['Class'].value_counts()
```

Class	count
0.0	5970
1.0	3

dtype: int64

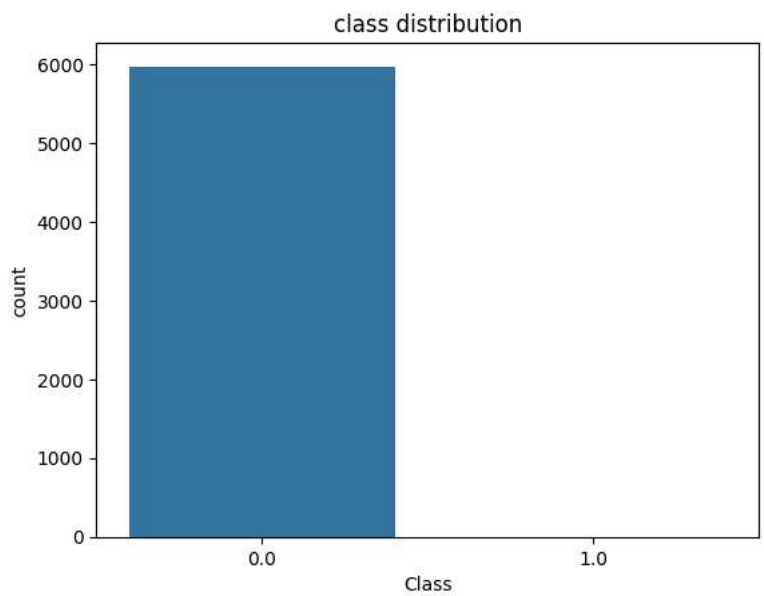
```
# get percentages for the class  which is important for imbalanced data
data['Class'].value_counts(normalize=True)*100
```

Class	proportion
0.0	99.949774
1.0	0.050226

dtype: float64

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
sns.countplot(x="Class",data=data)
plt.title("class distribution")
plt.show()
```




```
from sklearn.preprocessing import StandardScaler
scalar=StandardScaler() ##the obejct
data[["Time","Amount"]]=scalar.fit_transform(data[["Time","Amount"]])
```

```
data['Hour']=(data['Time']//3600)%24
data["LogAmount"]=np.log1p(data['Amount'])
```

```
data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V23	V24	V2
0	-1.517306	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.110474	0.066928	0.12853
1	-1.517306	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	0.101288	-0.339846	0.16717
2	-1.516740	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.909412	-0.689281	-0.32764
3	-1.516740	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.190321	-1.175575	0.64737
4	-1.516173	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.137458	0.141267	-0.20601

5 rows × 33 columns

```
from sklearn.model_selection import train_test_split
x= data.drop(['Class'], axis=1)
y = data['Class']

x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.3, stratify=y, random_state=42
)
```

```
from imblearn.over_sampling import SMOTE
```

```
# Reduce k_neighbors to avoid needing 6+ samples
smote =SMOTE(k_neighbors=1, random_state=42)
x_train_resampled, y_train_resampled = smote.fit_resample(x_train, y_train)
```

```
from collections import Counter
print("before SMOTE:",Counter(y_train))
print("after SMOTE:",Counter(y_train_resampled))
```

```
before SMOTE: Counter({0.0: 4179, 1.0: 2})
after SMOTE: Counter({0.0: 4179, 1.0: 4179})
```

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.11/dist-packages (2.1.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from xgboost) (2.0.2)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.11/dist-packages (from xgboost) (2.21.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from xgboost) (1.15.3)
```

```
import xgboost as xgb
from sklearn.metrics import classification_report,confusion_matrix,roc_auc_score, roc_curve
import matplotlib.pyplot as plt
```

```
#initialize and train XGBOOST
xgb_model = xgb.XGBClassifier(
    n_estimators=100,
    max_depth=5,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)

xgb_model.fit(x_train_resampled, y_train_resampled)
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [05:57:24] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.1, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=5,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=100,
              n_jobs=None, num_parallel_tree=None, random_state=42, ...)
```

```
##predict on original test set
y_pred = xgb_model.predict(x_test)
y_proba = xgb_model.predict_proba(x_test)[:, 1]  # for ROC AUC
```

```
print(y_pred)
print(y_proba)
```

Show hidden output

```
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nROC AUC Score:")
print(roc_auc_score(y_test, y_proba))
```

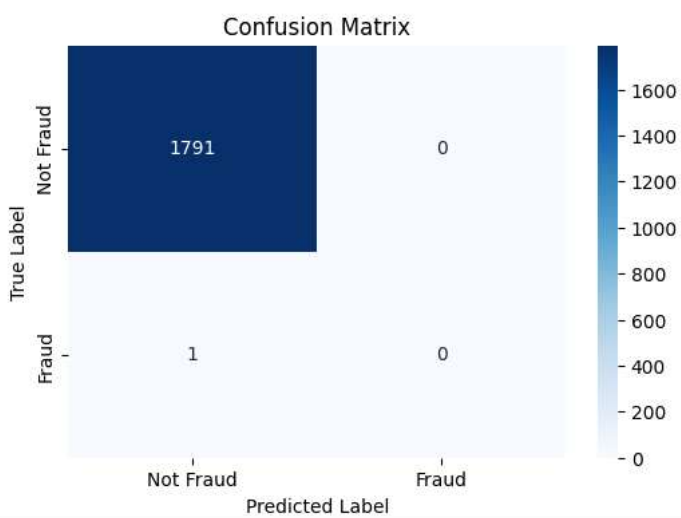
```
Confusion Matrix:
[[1791    0]
 [   1    0]]

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1791
         1.0       0.00      0.00      0.00         1

    accuracy                           1.00      1792
   macro avg       0.50      0.50      0.50      1792
weighted avg       1.00      1.00      1.00      1792


ROC AUC Score:
0.9681742043551089
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import classification_report, roc_auc_score

# Define base model
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)

# Define hyperparameter grid
param_dist = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}

# Setup RandomizedSearchCV
random_search = RandomizedSearchCV(
    estimator=xgb,
    param_distributions=param_dist,
    n_iter=20,
    scoring='roc_auc',
    cv=3,
    verbose=2,
    random_state=42,
    n_jobs=-1
)

# Fit on training data
random_search.fit(x_train_resampled, y_train_resampled)

# Predict on test set
best_model = random_search.best_estimator_
y_pred = best_model.predict(x_test)
y_proba = best_model.predict_proba(x_test)[:, 1]

# Evaluate
```

```
print("Classification Report:\n", classification_report(y_test, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test, y_proba))
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [05:58:04] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Classification Report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1791
         1.0       0.00      0.00      0.00         1

    accuracy                           1.00      1792
   macro avg       0.50      0.50      0.50      1792
weighted avg       1.00      1.00      1.00      1792

ROC AUC Score: 0.9625907314349527
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Get predictions
y_pred = best_model.predict(x_test)

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Not Fraud", "Fraud"], yticklabels=["Not Fraud", "Fraud"])
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```




```
!pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.8-cp311-cp311-manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: graphviz in /usr/local/lib/python3.11/dist-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<3.0,>=1.16.0 in /usr/local/lib/python3.11/dist-packages (from catboost) (2.0.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.11/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from catboost) (1.15.3)
```

```
Requirement already satisfied: plotly in /usr/local/lib/python3.11/dist-packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from catboost) (1.17.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost) (2.9.0.pc
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost) (4.58.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->catboost) (3.2.3)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly->catboost) (9.1.2)
Downloading catboost-1.2.8-cp311-cp311-manylinux2014_x86_64.whl (99.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.2/99.2 MB 9.2 MB/s eta 0:00:00
Installing collected packages: catboost
Successfully installed catboost-1.2.8
```

```
from catboost import CatBoostClassifier
from sklearn.metrics import classification_report, roc_auc_score

# Train CatBoost model (without hyperparameter tuning)
cat_model = CatBoostClassifier(verbose=0, random_state=42)
cat_model.fit(x_train_resampled, y_train_resampled)

# Predict
y_pred = cat_model.predict(x_test)
y_proba = cat_model.predict_proba(x_test)[:, 1]

# Evaluate
print("Classification Report:\n", classification_report(y_test, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test, y_proba))
```

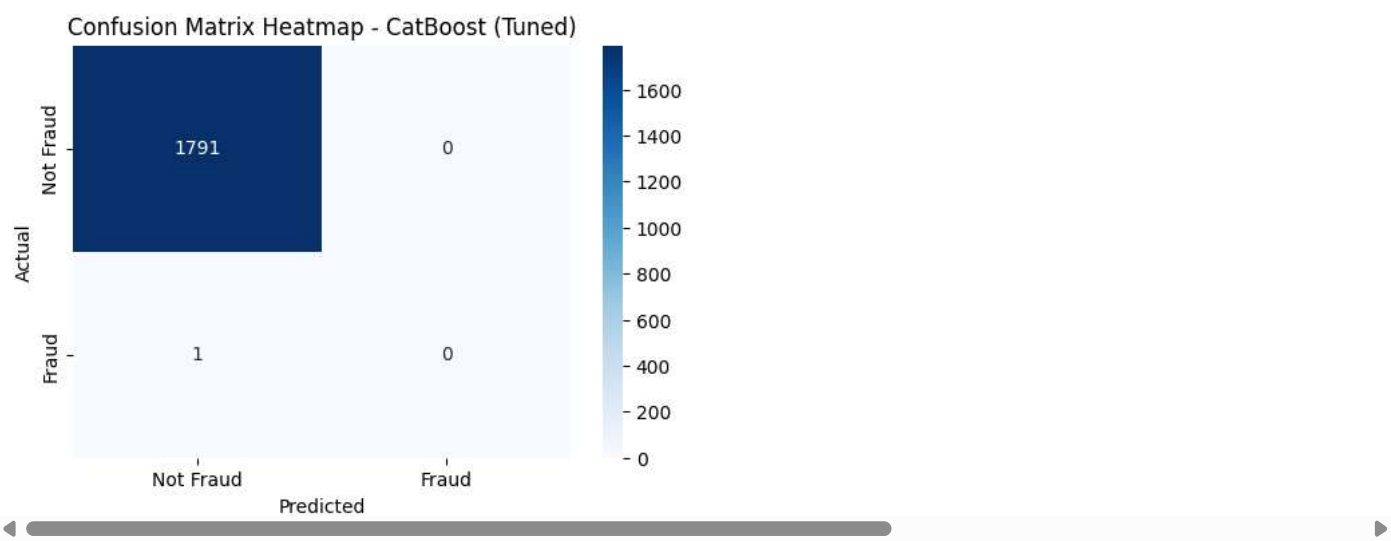
```
Classification Report:
                   precision    recall  f1-score   support

              0.0       1.00      1.00      1.00      1791
              1.0       0.00      0.00      0.00         1

         accuracy                           1.00      1792
        macro avg       0.50      0.50      0.50      1792
     weighted avg       1.00      1.00      1.00      1792

ROC AUC Score: 0.9570072585147963
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
from catboost import CatBoostClassifier
from sklearn.model_selection import RandomizedSearchCV

# 1. Define the model
cat_model = CatBoostClassifier(verbose=0, random_state=42)

# 2. Define the parameter grid for tuning
param_dist = {
    'iterations': [100, 200, 300],
    'depth': [4, 6, 8],
    'learning_rate': [0.01, 0.05, 0.1],
    'l2_leaf_reg': [1, 3, 5, 7],
    'border_count': [32, 64, 128]
}

# 3. Set up RandomizedSearchCV
cat_random_search = RandomizedSearchCV(
    estimator=cat_model,
    param_distributions=param_dist,
    n_iter=20,                  # Number of random combinations
    scoring='roc_auc',
    cv=3,
    verbose=2,
    random_state=42,
    n_jobs=-1
```

```
)

# 4. Fit the model to resampled (SMOTE) training data
cat_random_search.fit(x_train_resampled, y_train_resampled)

# 5. Best model
best_cat_model = cat_random_search.best_estimator_

# 6. Evaluate on original test data
from sklearn.metrics import classification_report, roc_auc_score

y_pred_cat = best_cat_model.predict(x_test)
y_proba_cat = best_cat_model.predict_proba(x_test)[:, 1]

print("\nClassification Report:\n", classification_report(y_test, y_pred_cat))
print("ROC AUC Score:", roc_auc_score(y_test, y_proba_cat))
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits

Classification Report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1791
         1.0       0.00      0.00      0.00         1

    accuracy                           1.00      1792
   macro avg       0.50      0.50      0.50      1792
weighted avg       1.00      1.00      1.00      1792

ROC AUC Score: 0.9977666108319375
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# 1. Get predictions again (if needed)
y_pred_cat = best_cat_model.predict(x_test)

# 2. Create confusion matrix
cm = confusion_matrix(y_test, y_pred_cat)

# 3. Plot heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Fraud', 'Fraud'], yticklabels=['Not Fraud', 'Fraud'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap - CatBoost (Tuned)')
plt.show()
```




```
!pip install shap
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packages (0.47.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from shap) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from shap) (1.15.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from shap) (1.6.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from shap) (2.2.2)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.11/dist-packages (from shap) (4.67.1)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.11/dist-packages (from shap) (24.2)
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.11/dist-packages (from shap) (0.0.8)
Requirement already satisfied: numba>=0.54 in /usr/local/lib/python3.11/dist-packages (from shap) (0.60.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.11/dist-packages (from shap) (3.1.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from shap) (4.14.0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba>=0.54->shap) (0.43.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->shap) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->shap) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->shap) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->shap) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->shap) (3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->shap) (1.17.0)
```

```
import shap


shap.initjs()

# 4. Create SHAP TreeExplainer (CatBoost is tree-based)
explainer = shap.TreeExplainer(best_cat_model)

# 5. Calculate SHAP values for the test set
shap_values = explainer.shap_values(x_test)

# 6. Summary Plot – Global Feature Importance
shap.summary_plot(shap_values, x_test)

# 7. Bar Plot – Sorted global feature importance
shap.summary_plot(shap_values, x_test, plot_type="bar")

# 8. Force Plot for a single prediction (index = 0, you can change it)
sample_index = 0
shap.force_plot(
    explainer.expected_value,
    shap_values[sample_index],
    x_test.iloc[sample_index],
    matplotlib=True
)
```