Machine Learning Project - Credit Card Fraud Detection

Name: Vahid Sabri Candan

Data does not contain any text and column names looks correct so no need to deal with thoose.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, AdaBoostClassifier
from sklearn.preprocessing import RobustScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, StratifiedKFold,
GridSearchCV
from sklearn.metrics import average_precision_score, recall_score,
precision_score, f1_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
import optuna
from optuna.pruners import MedianPruner
from optuna.samplers import TPESampler
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow as tf
import warnings
```

```
d:\projects_last\.venv\Lib\site-packages\tqdm\auto.py:21:
TqdmWarning: IProgress not found. Please update jupyter and
ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [2]:
```
df = pd.read_csv("creditcard.csv")
df.describe()
```

Out[2]:

	Time	V1	V2	V3
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	1.175161e-15	3.384974e-16	-1.379537e-15
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00

8 rows × 31 columns

No empty rows

In [3]:
```
print(df.info())
p = df.isnull().sum()
p = p.sort_values()
print(p)
```

```
<class 'pandas.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
None
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
```

Exploratory statistical analysis

In [4]:
```
print(df[["Time", "Amount", "Class"]].describe())
```

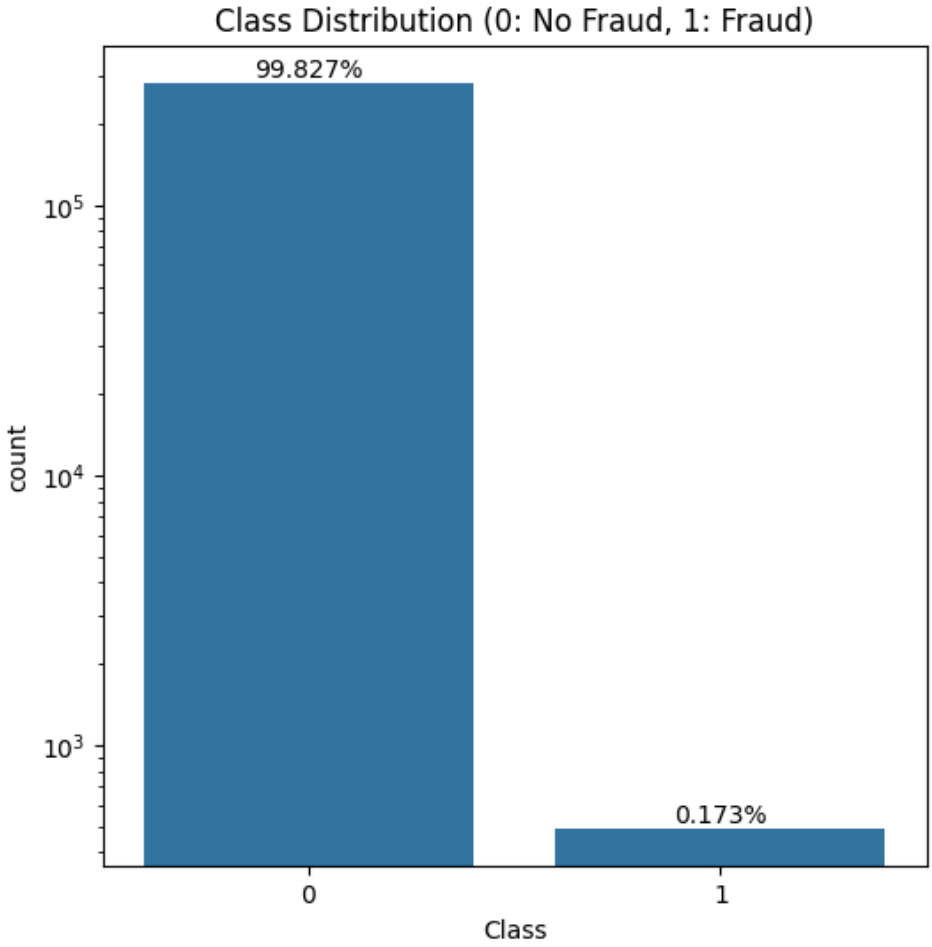
```
                Time         Amount          Class
count  284807.000000  284807.000000  284807.000000
mean    94813.859575      88.349619       0.001727
std     47488.145955     250.120109       0.041527
min         0.000000       0.000000       0.000000
25%     54201.500000       5.600000       0.000000
50%     84692.000000      22.000000       0.000000
75%    139320.500000      77.165000       0.000000
max    172792.000000   25691.160000       1.000000
```

Class based

Data is pretty unbalanced as we can see, with frauds accounting for only 0.173% of all transactions.

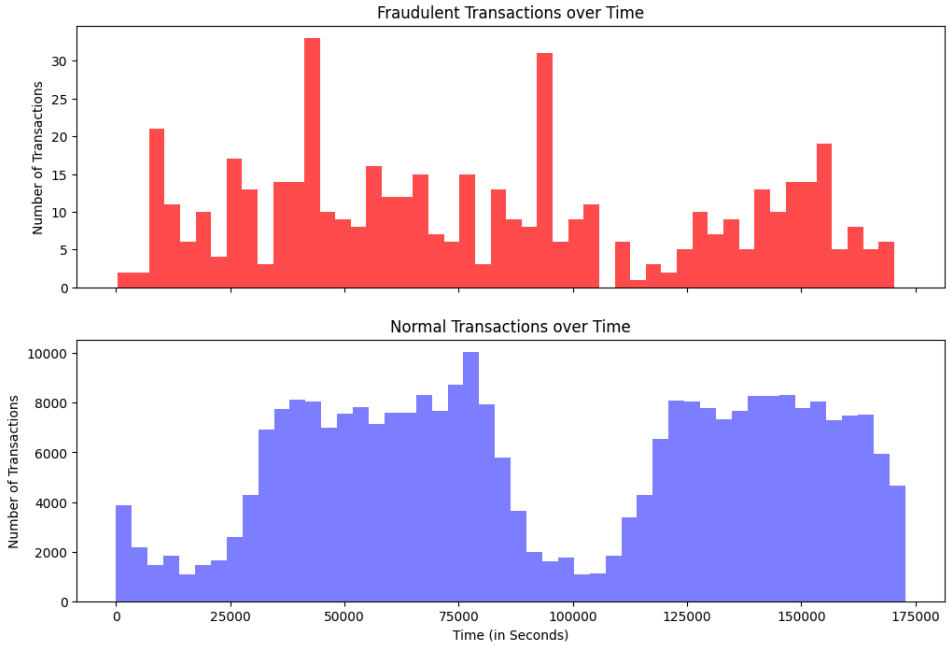
This severe skewness means that standard accuracy is a meaningless metric. A model predicting "No Fraud" for every transaction would achieve 99.8% accuracy but fail to detect any fraud. We must prioritize metrics like Area Under the Precision-Recall Curve (AUPRC).

In [5]:
```
plt.figure(figsize=(6, 6))
ax = sns.countplot(x="Class", data=df)
plt.title("Class Distribution (0: No Fraud, 1: Fraud)")
plt.yscale("log") # Log scale to see better
# calculation of percentages
total = len(df)
for p in ax.patches:
    percentage = "{:.3f}%".format(100 * p.get_height()/total)
    x = p.get_x() + p.get_width()/2
    y = p.get_height()
    ax.annotate(percentage, (x, y), ha="center", va="bottom")
plt.show()
```



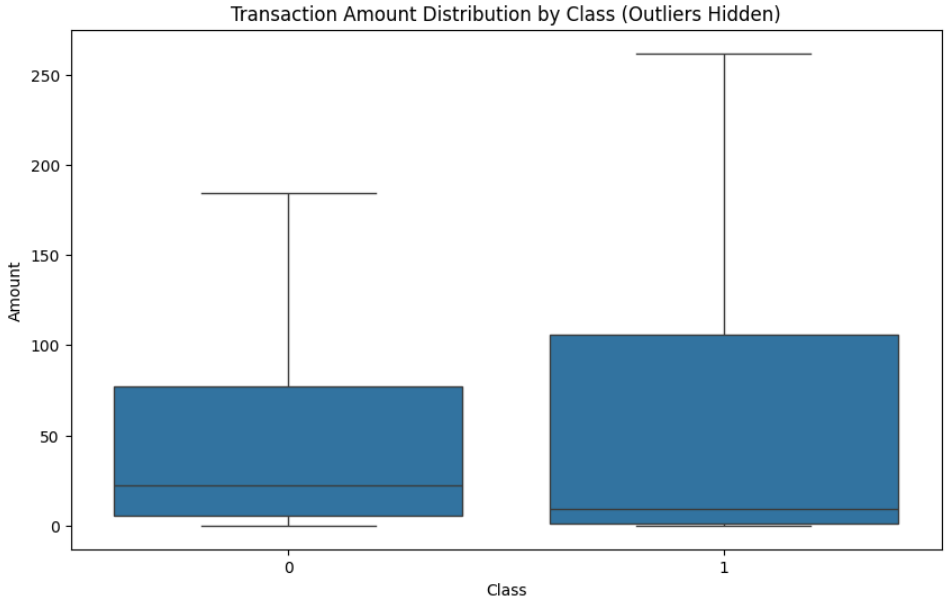
Time based

In [6]:
```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(12, 8))

ax1.hist(df.Time[df.Class == 1], bins=50, color="red", alpha=0.7)
ax1.set_title("Fraudulent Transactions over Time")
ax1.set_ylabel("Number of Transactions")

ax2.hist(df.Time[df.Class == 0], bins=50, color="blue", alpha=0.5)
ax2.set_title("Normal Transactions over Time")
ax2.set_xlabel("Time (in Seconds)")
ax2.set_ylabel("Number of Transactions")
plt.show()
```



Amount based

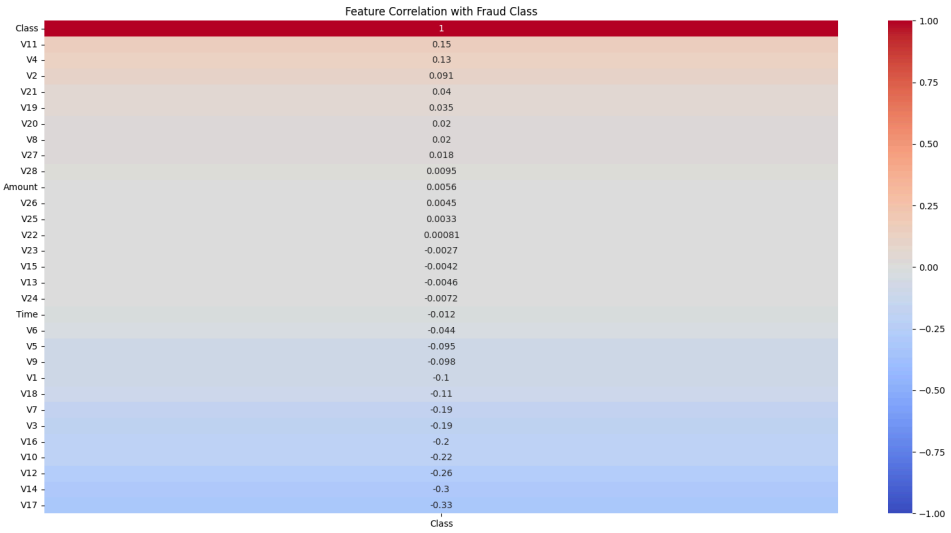
In [7]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x="Class", y="Amount", data=df, showfliers=False) # Hiding
extreme outliers for clarity
plt.title("Transaction Amount Distribution by Class (Outliers Hidden)")
plt.show()
```



Correlation

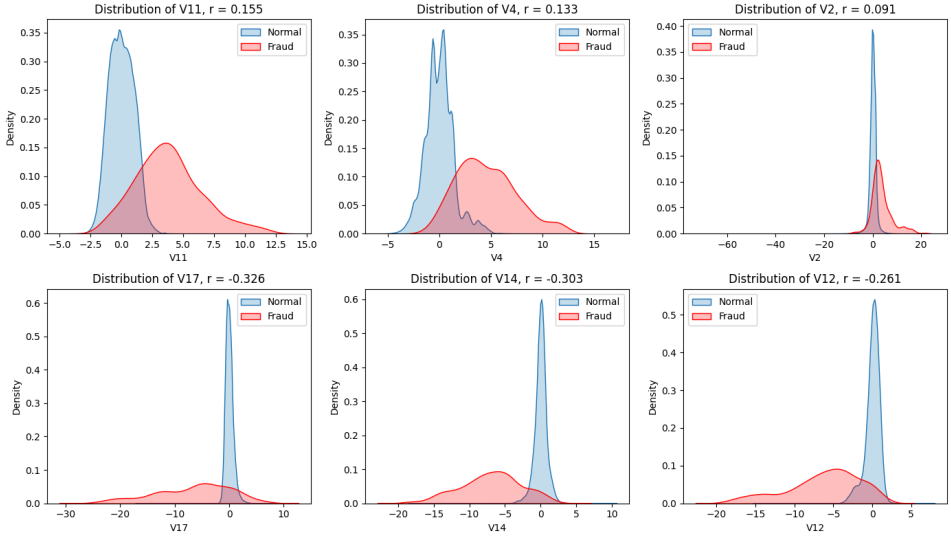
this chart actually not that usefull in this case because most of correlations close to 0

In [8]:
```
plt.figure(figsize=(20, 10))
# Calculate correlation
corr = df.corr()
# We focus on correlations with "Class" specifically
sns.heatmap(corr[["Class"]].sort_values(by="Class", ascending=False),
            annot=True, cmap="coolwarm", vmin=-1, vmax=1)
plt.title("Feature Correlation with Fraud Class")
plt.show()
```

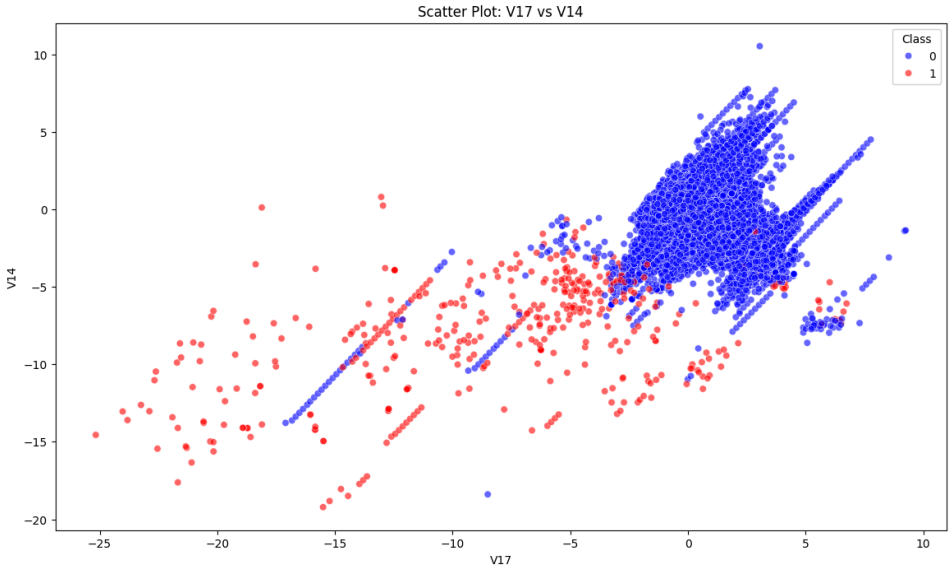


we take the most correlated or negatively correlated categories

In [9]:
```
correlations = df.corrwith(df["Class"]).sort_values(ascending=False)
features_to_check =
correlations.drop("Class").nlargest(3).index.tolist() +
correlations.drop("Class").nsmallest(3).index.tolist()

plt.figure(figsize=(14, 8))
for i, col in enumerate(features_to_check, 1):
    plt.subplot(2, 3, i)
    sns.kdeplot(df[df["Class"] == 0][col], label="Normal", fill=True)
    sns.kdeplot(df[df["Class"] == 1][col], label="Fraud", fill=True,
color="red")
    plt.title(f"Distribution of {col}, r = {(correlations[col]):.3f}")
    plt.legend()
plt.tight_layout()
plt.show()
```



In [10]:
```
plt.figure(figsize=(14, 8))
sns.scatterplot(x="V17", y="V14", hue="Class", data=df, alpha=0.6,
palette={0:"blue", 1:"red"})
plt.title("Scatter Plot: V17 vs V14")
plt.show()
```

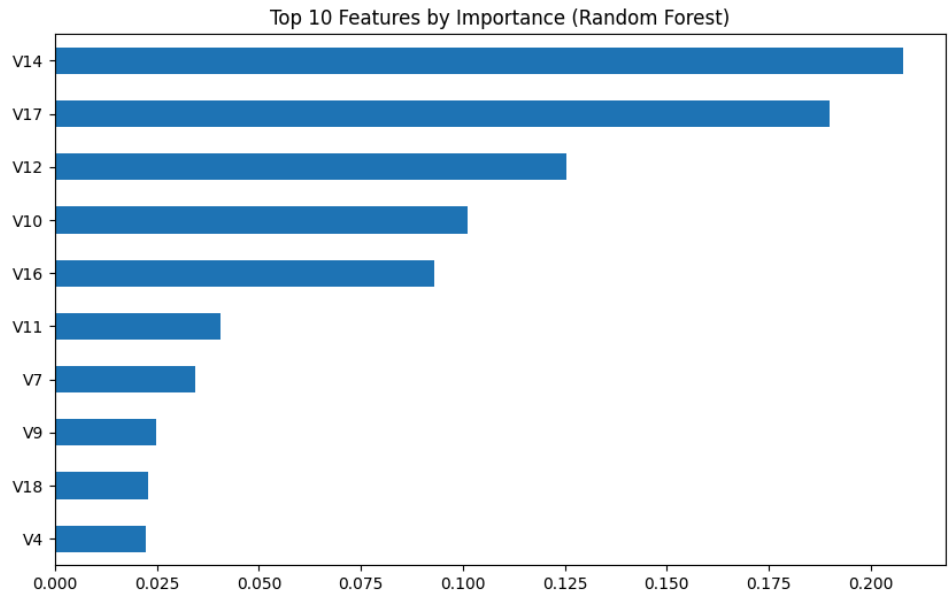


Features V17, V14, and V12 show the strongest negative correlation with the Class variable.

The scatter plot of V17 vs V14 reveals a clear separation between fraud and normal transactions for highly negative values. This suggests that these PCA-transformed features contain significant seperated power, likely showing specific types of fraudulent spending patterns.

Feature importances with random forest algoritm

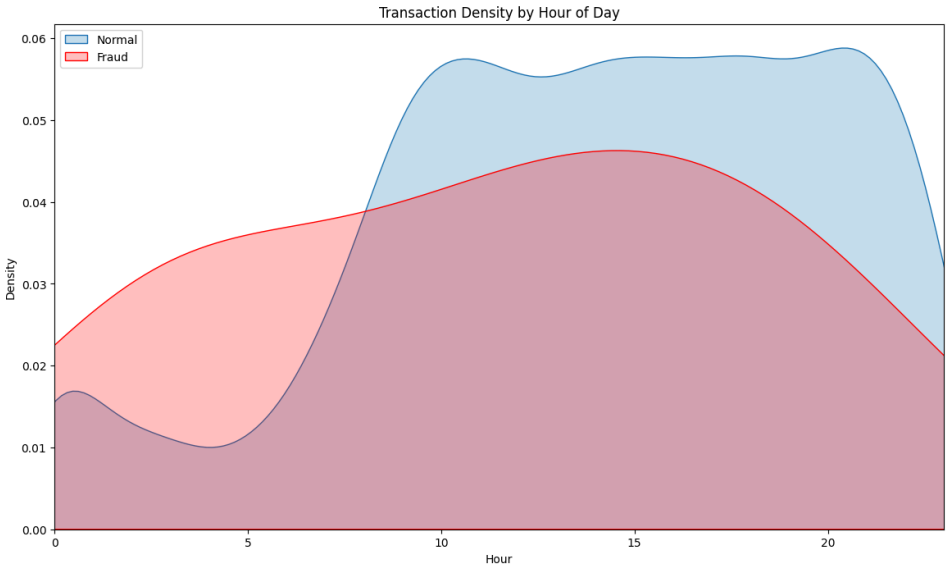
In [11]:
```
X = df.drop("Class", axis=1)
y = df["Class"]

# Initialize a small model just for importance extraction
rf = RandomForestClassifier(n_estimators=50, max_depth=5,
random_state=42, n_jobs=-1)
rf.fit(X, y)

# Plotting importance
importances = pd.Series(rf.feature_importances_,
index=X.columns).sort_values(ascending=False).head(10)
plt.figure(figsize=(10, 6))
importances.plot(kind="barh", title="Top 10 Features by Importance
(Random Forest)")
plt.gca().invert_yaxis()
plt.show()
```



Time of the day based analysis

One thing is pretty surprising, i would expect that frauds to concentrate in the night hours.

In [12]:
```
df["Hour"] = df["Time"].apply(lambda x: np.floor(x / 3600) % 24)

plt.figure(figsize=(14, 8))

# Plot the distribution of Fraud vs Normal transactions across hours
sns.kdeplot(df[df["Class"] == 0]["Hour"], label="Normal", fill=True,
bw_adjust=2)
sns.kdeplot(df[df["Class"] == 1]["Hour"], label="Fraud", fill=True,
color="red", bw_adjust=2)

plt.title("Transaction Density by Hour of Day")
plt.xlabel("Hour")
plt.ylabel("Density")
plt.legend()
plt.xlim(0, 23)
plt.show()
```



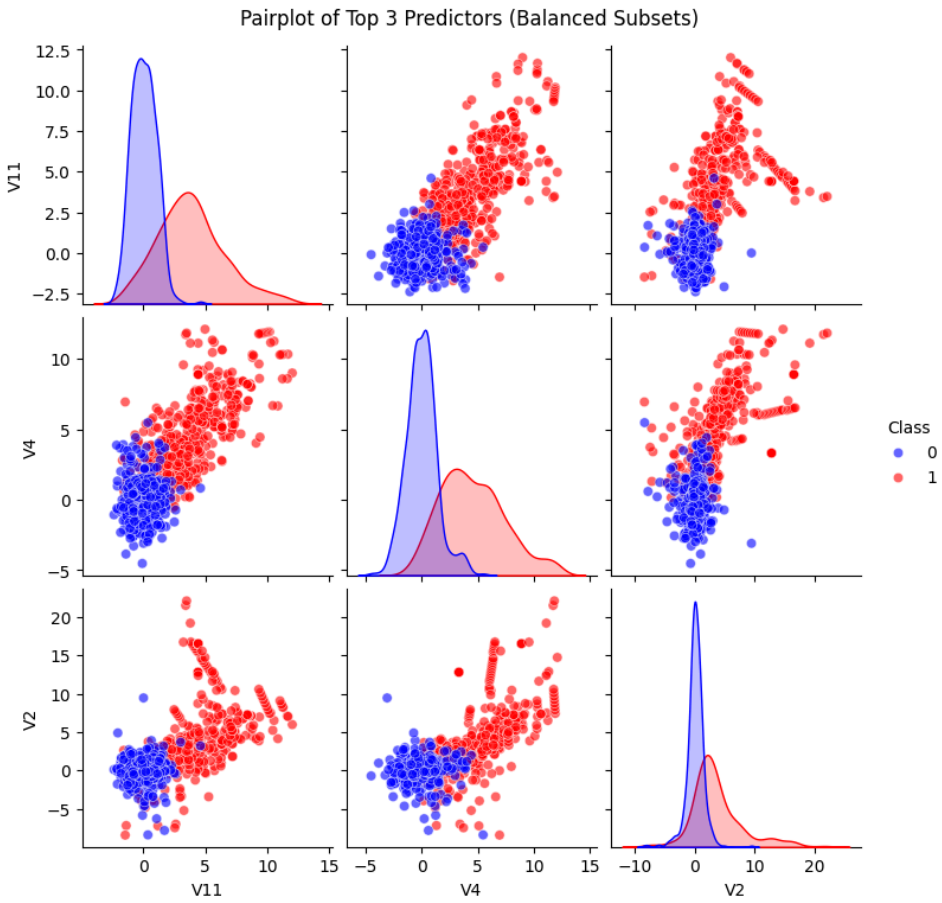
Subsets against each other

This is to see if we can catch any clustering

In [ ]:
```
top_features = correlations.drop("Class").nlargest(3).index.tolist()
print(f"Top 3 Features selected for Pairplot: {top_features}")


fraud_df = df[df["Class"] == 1]
normal_df = df[df["Class"] == 0].sample(n=len(fraud_df),
random_state=42)
plot_df = pd.concat([fraud_df, normal_df])

# 3. Create the Pairplot
plt.figure(figsize=(14, 8))
pp = sns.pairplot(plot_df,
                  vars=top_features,
                  hue="Class",
                  palette={0: "blue", 1: "red"},
                  kind="scatter",
                  diag_kind="kde",
                  plot_kws={"alpha": 0.6})

pp.fig.suptitle("Pairplot of Top 3 Predictors (Balanced Subsets)",
y=1.02)
plt.show()
```

```
Top 3 Features selected for Pairplot: ['V11', 'V4', 'V2']
```

```
<Figure size 1400x800 with 0 Axes>
```



Preprocessing

We already checked for missing data shortly after we imported our dataset and dataset does not include any missing row or data.

For duplicate data

In [14]:
```
befor_dup = len(df)
df_clean = df.drop_duplicates()
after_dup = len(df_clean)
print(f"Number of duplicates: {befor_dup - after_dup}")
```

```
Number of duplicates: 1081
```

Feature scaling

In [ ]:
```
scaler = RobustScaler()

df_clean["scaled_amount"] =
scaler.fit_transform(df_clean["Amount"].values.reshape(-1,1))
df_clean["scaled_time"] =
scaler.fit_transform(df_clean["Time"].values.reshape(-1,1))
df_clean["log_amount"] = np.log(df_clean["scaled_amount"] + 1e-9 +
abs(df_clean["scaled_amount"].min()))


df_clean.drop(["Time", "Amount"], axis=1, inplace=True)
```

Outlier Detection
I prefer to not remove the outlier because it might be part of the information encoded on PCA transformed data

In [16]:
```
feature_list_sorted = correlations.drop(["Class", "Amount",
"Time"]).index.tolist()
print(f"Outlier Detection Targets (sorted via r-scores):
{feature_list_sorted}.")

# Quartiles
Q1 = df_clean[feature_list_sorted].quantile(0.25)
Q3 = df_clean[feature_list_sorted].quantile(0.75)
IQR = Q3 - Q1

# Bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outlier_count = ((df_clean[feature_list_sorted] < lower_bound) |
(df_clean[feature_list_sorted] > upper_bound)).sum()

for col, count in outlier_count.items():
    print(f" - {col}: {count} outliers detected")
```

```
Outlier Detection Targets (sorted via r-scores): ['V11', 'V4',
'V2', 'V21', 'V19', 'V20', 'V8', 'V27', 'V28', 'V26', 'V25',
'V22', 'V23', 'V15', 'V13', 'V24', 'V6', 'V5', 'V9', 'V1', 'V18',
'V7', 'V3', 'V16', 'V10', 'V12', 'V14', 'V17'].
 - V11: 735 outliers detected
 - V4: 11094 outliers detected
 - V2: 13390 outliers detected
 - V21: 14401 outliers detected
 - V19: 10150 outliers detected
 - V20: 27553 outliers detected
 - V8: 23904 outliers detected
 - V27: 38799 outliers detected
 - V28: 30094 outliers detected
 - V26: 5665 outliers detected
 - V25: 5333 outliers detected
 - V22: 1298 outliers detected
 - V23: 18467 outliers detected
 - V15: 2884 outliers detected
 - V13: 3362 outliers detected
 - V24: 4758 outliers detected
 - V6: 22886 outliers detected
 - V5: 12221 outliers detected
 - V9: 8199 outliers detected
 - V1: 6948 outliers detected
 - V18: 7468 outliers detected
 - V7: 8839 outliers detected
 - V3: 3306 outliers detected
 - V16: 8180 outliers detected
 - V10: 9345 outliers detected
 - V12: 15282 outliers detected
 - V14: 14060 outliers detected
 - V17: 7353 outliers detected
```

In [17]:
```
X = df_clean.drop("Class", axis = 1) # Model inputs
y = df_clean.Class   # Model outputs

X_val_train, X_test, y_val_train, y_test = train_test_split(X, y,
test_size=0.1, random_state=13, stratify=y)

X_train, X_val, y_train, y_val = train_test_split(
    X_val_train, y_val_train,
    test_size=1/9,
    random_state=13,
    shuffle=True
)

print(f"Train shape {X_train.shape}, Test shape {X_test.shape},
Validation shape {X_val.shape}")
print(f"Train set Non-Fraud Ratio:
{y_train.value_counts(normalize=True)[0]:.4%}")
print(f"Test set Non-Fraud Ratio:
{y_test.value_counts(normalize=True)[0]:.4%}")
print(f"Validation set Non-Fraud Ratio:
{y_val.value_counts(normalize=True)[0]:.4%}")
```

```
Train shape (226980, 32), Test shape (28373, 32), Validation shape
(28373, 32)
Train set Non-Fraud Ratio:       99.8291%
Test set Non-Fraud Ratio:        99.8343%
Validation set Non-Fraud Ratio:  99.8661%
```

The class imbalance in this dataset is significant (0.17% fraud). A naive model that predicts only the majority class (y=0) would achieve 99% accuracy, but it would fail completely at the primary objective: detecting fraud (Recall = 0).

Therefore, standard algorithms like Logistic Regression will be ineffective unless adapted. To train a useful model, we cannot rely on accuracy; we must optimize for Recall and AUPRC. Furthermore, we must implement specific strategies to overcome this bias, such as Resampling or Cost-Sensitive Learning, rather than just relying on more complex algorithms.

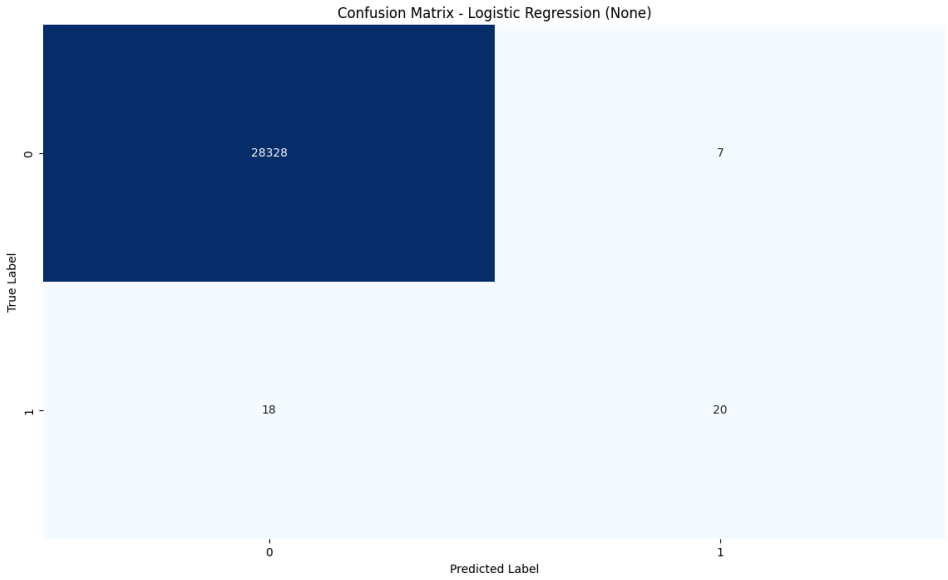
Model evaluator code:

In [18]:

```
results = []


def evaluate_model(name, model, X_tr, y_tr, sampling_method,
with_graph=False, X_val=X_val, y_val=y_val):
    # Train
    model.fit(X_tr, y_tr)

    # Predict on Validation Set
    # We use predict_proba for AUPRC
    y_prob = model.predict_proba(X_val)[:, 1]
    y_pred = model.predict(X_val)

    # Calculate Metrics
    auprc = average_precision_score(y_val, y_prob)
    recall = recall_score(y_val, y_pred)
    precision = precision_score(y_val, y_pred)
    f1 = f1_score(y_val, y_pred)

    # Log results
    results.append({
        "Model": name,
        "Sampling Technique": sampling_method,
        "AUPRC": auprc,
        "Recall": recall,
        "Precision": precision,
        "F1 Score": f1
    })

    # Generate confusion matrix
    cm = confusion_matrix(y_val, y_pred)
    if with_graph:
        plt.figure(figsize=(14, 8))
        sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
        plt.title(f"Confusion Matrix - {name} ({sampling_method})")
        plt.ylabel("True Label")
        plt.xlabel("Predicted Label")
        plt.show()

    return classification_report(y_val, y_pred)
```

Logistic regression models

In [19]:
```
model1 = LogisticRegression(max_iter=1000)
model1_results = evaluate_model("Logistic Regression", model1, X_train,
y_train, "None", with_graph=True)
print(model1_results)
```



	precision	recall	f1-score	support
0	1.00	1.00	1.00	28335
1	0.74	0.53	0.62	38
accuracy			1.00	28373
macro avg	0.87	0.76	0.81	28373
weighted avg	1.00	1.00	1.00	28373

In [20]:
```
for _ in [0.1, 1, 6, 10, 100, 1000]:
    print(f"Class weights: {{0:1, 1:{_}}}")
    model1_with_weights = LogisticRegression(max_iter=1000,
class_weight={0:1, 1:_})
    model1_weights_results = evaluate_model("Logistic Regression",
model1_with_weights, X_train, y_train, f"Class Weights{{0:1, 1:{_}}}")
    print(model1_weights_results)
```

```
Class weights: {0:1, 1:0.1}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.78      0.18      0.30        38

    accuracy                           1.00     28373
   macro avg       0.89      0.59      0.65     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:1}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.74      0.53      0.62        38

    accuracy                           1.00     28373
   macro avg       0.87      0.76      0.81     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:6}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.72      0.89      0.80        38

    accuracy                           1.00     28373
   macro avg       0.86      0.95      0.90     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:10}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.71      0.89      0.79        38

    accuracy                           1.00     28373
   macro avg       0.85      0.95      0.90     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:100}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.23      0.97      0.38        38

    accuracy                           1.00     28373
   macro avg       0.62      0.98      0.69     28373
```

```
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:1000}
              precision    recall  f1-score   support

           0       1.00      0.95      0.98     28335
           1       0.03      1.00      0.05        38

    accuracy                           0.95     28373
   macro avg       0.51      0.98      0.52     28373
weighted avg       1.00      0.95      0.97     28373
```

SMOTE"d dataset

In [21]:
```
X_train_smote, y_train_smote =
SMOTE(random_state=13).fit_resample(X_train, y_train)
```

In [22]:
```
for _ in [0.1, 1, 6, 10, 100, 1000]:
    print(f"Class weights: {{0:1, 1:{_}}}")
    model1_with_weights = LogisticRegression(max_iter=1000,
class_weight={0:1, 1:_})
    model1_weights_results = evaluate_model("Logistic Regression",
model1_with_weights, X_train_smote, y_train_smote, f"SMOTE'd/Class
Weights{{0:1, 1:{_}}}")
    print(model1_weights_results)
```

```
Class weights: {0:1, 1:0.1}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.37      0.97      0.54        38

    accuracy                           1.00     28373
   macro avg       0.68      0.99      0.77     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:1}
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     28335
           1       0.05      1.00      0.09        38

    accuracy                           0.97     28373
   macro avg       0.52      0.99      0.54     28373
weighted avg       1.00      0.97      0.99     28373

Class weights: {0:1, 1:6}
              precision    recall  f1-score   support

           0       1.00      0.90      0.95     28335
           1       0.01      1.00      0.03        38

    accuracy                           0.90     28373
   macro avg       0.51      0.95      0.49     28373
weighted avg       1.00      0.90      0.94     28373

Class weights: {0:1, 1:10}
              precision    recall  f1-score   support

           0       1.00      0.86      0.92     28335
           1       0.01      1.00      0.02        38

    accuracy                           0.86     28373
   macro avg       0.50      0.93      0.47     28373
weighted avg       1.00      0.86      0.92     28373

Class weights: {0:1, 1:100}
              precision    recall  f1-score   support

           0       1.00      0.68      0.81     28335
           1       0.00      1.00      0.01        38

    accuracy                           0.68     28373
```

```
   macro avg       0.50      0.84      0.41     28373
weighted avg       1.00      0.68      0.81     28373

Class weights: {0:1, 1:1000}
              precision    recall  f1-score   support

           0       1.00      0.50      0.67     28335
           1       0.00      1.00      0.01        38

    accuracy                           0.50     28373
   macro avg       0.50      0.75      0.34     28373
weighted avg       1.00      0.50      0.67     28373
```

In [23]:
```
X_train_under, y_train_under =
RandomUnderSampler(random_state=13).fit_resample(X_train, y_train)
```

In [24]:
```
for _ in [0.1, 1, 6, 10, 100, 1000]:
    print(f"Class weights: {{0:1, 1:{_}}}")
    model1_with_weights = LogisticRegression(max_iter=1000,
class_weight={0:1, 1:_})
    model1_weights_results = evaluate_model("Logistic Regression",
model1_with_weights, X_train_under, y_train_under, f"Under-sampled/Class
Weights{{0:1, 1:{_}}}")
    print(model1_weights_results)
```

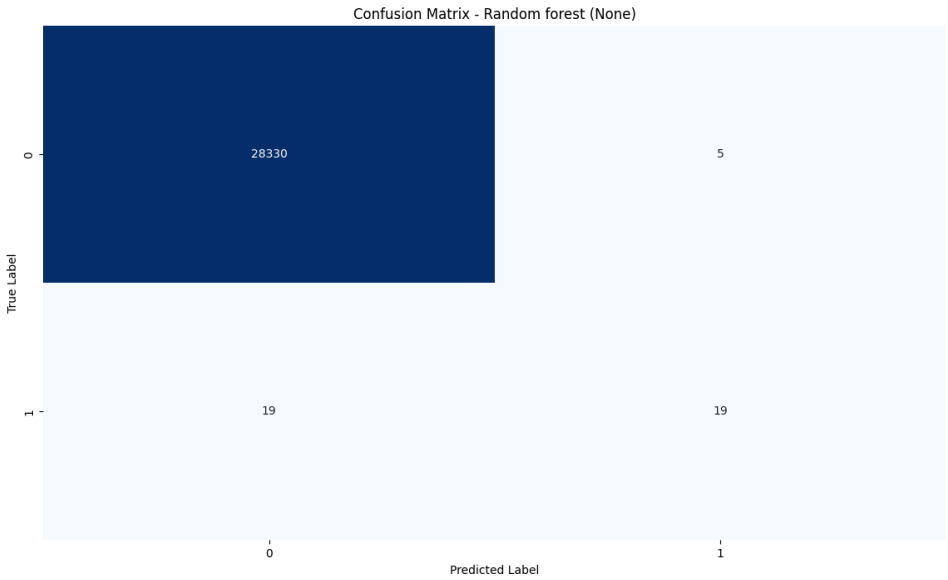
```
Class weights: {0:1, 1:0.1}
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     28335
           1       0.08      0.97      0.15        38

    accuracy                           0.99     28373
   macro avg       0.54      0.98      0.57     28373
weighted avg       1.00      0.99      0.99     28373

Class weights: {0:1, 1:1}
              precision    recall  f1-score   support

           0       1.00      0.95      0.97     28335
           1       0.02      1.00      0.05        38

    accuracy                           0.95     28373
   macro avg       0.51      0.97      0.51     28373
weighted avg       1.00      0.95      0.97     28373

Class weights: {0:1, 1:6}
              precision    recall  f1-score   support

           0       1.00      0.86      0.92     28335
           1       0.01      1.00      0.02        38

    accuracy                           0.86     28373
   macro avg       0.50      0.93      0.47     28373
weighted avg       1.00      0.86      0.92     28373

Class weights: {0:1, 1:10}
              precision    recall  f1-score   support

           0       1.00      0.82      0.90     28335
           1       0.01      1.00      0.01        38

    accuracy                           0.82     28373
   macro avg       0.50      0.91      0.46     28373
weighted avg       1.00      0.82      0.90     28373

Class weights: {0:1, 1:100}
              precision    recall  f1-score   support

           0       1.00      0.66      0.80     28335
           1       0.00      1.00      0.01        38

    accuracy                           0.66     28373
```

```
   macro avg       0.50      0.83      0.40     28373
weighted avg       1.00      0.66      0.80     28373

Class weights: {0:1, 1:1000}
              precision    recall  f1-score   support

           0       1.00      0.50      0.67     28335
           1       0.00      1.00      0.01        38

    accuracy                           0.50     28373
   macro avg       0.50      0.75      0.34     28373
weighted avg       1.00      0.50      0.67     28373
```

Random Forest Models

Plain:

In [25]:
```
model2 = RandomForestClassifier(n_estimators=100, max_depth=3,
random_state=13, n_jobs=-1)
model2_results = evaluate_model("Random forest", model2, X_train,
y_train, "None", with_graph=True)
print(model2_results)
```




```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.79      0.50      0.61        38

    accuracy                           1.00     28373
   macro avg       0.90      0.75      0.81     28373
weighted avg       1.00      1.00      1.00     28373
```

In [26]:
```
for _ in [0.1, 1, 10, 100, 1000]:
    print(f"Class weights: {{0:1, 1:{_}}}")
    model2_with_weights = RandomForestClassifier(n_estimators=100,
max_depth=3, random_state=13, n_jobs=-1, class_weight={0:1, 1:_})
    model2_weights_results = evaluate_model("Random forest",
model2_with_weights, X_train, y_train, f"Class Weights {{0:1, 1:{_}}}")
    print(model2_weights_results)
```

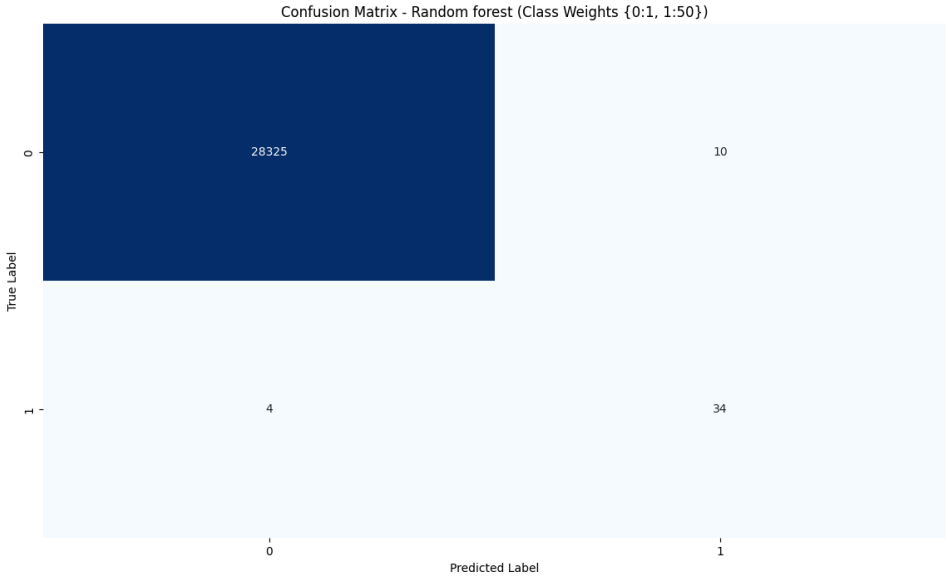
```
Class weights: {0:1, 1:0.1}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       1.00      0.29      0.45        38

    accuracy                           1.00     28373
   macro avg       1.00      0.64      0.72     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:1}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.79      0.50      0.61        38

    accuracy                           1.00     28373
   macro avg       0.90      0.75      0.81     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:10}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.81      0.79      0.80        38

    accuracy                           1.00     28373
   macro avg       0.91      0.89      0.90     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:100}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.77      0.95      0.85        38

    accuracy                           1.00     28373
   macro avg       0.88      0.97      0.92     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:1000}
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     28335
           1       0.10      0.97      0.19        38

    accuracy                           0.99     28373
   macro avg       0.55      0.98      0.59     28373
```

weighted avg	1.00	0.99	0.99	28373

Best one on the random forest

In [27]:
```
for _ in [50]:
    print(f"Class weights: {{0:1, 1:{_}}}")
    model2_with_weights = RandomForestClassifier(n_estimators=100,
max_depth=3, random_state=13, n_jobs=-1, class_weight={0:1, 1:_})
    model2_weights_results = evaluate_model("Random forest",
model2_with_weights, X_train, y_train, f"Class Weights {{0:1, 1:{_}}}",
with_graph=True)
    print(model2_weights_results)
```

```
Class weights: {0:1, 1:50}
```



	precision	recall	f1-score	support
0	1.00	1.00	1.00	28335
1	0.77	0.89	0.83	38
accuracy			1.00	28373
macro avg	0.89	0.95	0.91	28373
weighted avg	1.00	1.00	1.00	28373

In [28]:
```
for _ in [0.1, 1, 10, 100, 1000]:
    print(f"Class weights: {{0:1, 1:{_}}}")
    model2_smote = RandomForestClassifier(n_estimators=100, max_depth=3,
random_state=13, n_jobs=-1, class_weight={0:1, 1:_})
    model2_smote_results = evaluate_model("Random forest", model2_smote,
X_train_smote, y_train_smote, f"SMOTE'd/Class Weights {{0:1, 1:{_}}}")
    print(model2_smote_results)
```

```
Class weights: {0:1, 1:0.1}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.77      0.95      0.85        38

    accuracy                           1.00     28373
   macro avg       0.88      0.97      0.92     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:1}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.21      0.97      0.34        38

    accuracy                           1.00     28373
   macro avg       0.60      0.98      0.67     28373
weighted avg       1.00      1.00      1.00     28373

Class weights: {0:1, 1:10}
              precision    recall  f1-score   support

           0       1.00      0.71      0.83     28335
           1       0.00      1.00      0.01        38

    accuracy                           0.71     28373
   macro avg       0.50      0.86      0.42     28373
weighted avg       1.00      0.71      0.83     28373

Class weights: {0:1, 1:100}
              precision    recall  f1-score   support

           0       1.00      0.09      0.17     28335
           1       0.00      1.00      0.00        38

    accuracy                           0.09     28373
   macro avg       0.50      0.55      0.08     28373
weighted avg       1.00      0.09      0.17     28373

Class weights: {0:1, 1:1000}
              precision    recall  f1-score   support

           0       1.00      0.06      0.11     28335
           1       0.00      1.00      0.00        38

    accuracy                           0.06     28373
   macro avg       0.50      0.53      0.06     28373
```

```
weighted avg       1.00      0.06      0.11     28373
```

Gradient boost

it took so long i had to restart python kernel multiple times so i did not like this model

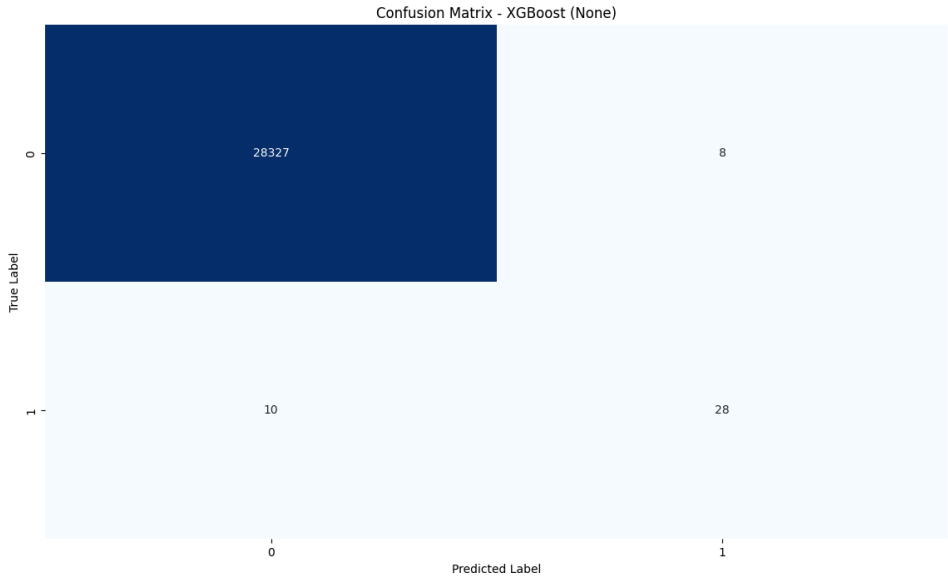
In [29]:
```
for _ in [1]:
    print(f"Class weights: {{0:1, 1:{_}}}")
    model3 = GradientBoostingClassifier(n_estimators=100,
random_state=13)
    model3_results = evaluate_model("Gradient Boosting", model3,
X_train, y_train, "None")
    print(model3_results)
```

```
Class weights: {0:1, 1:1}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.58      0.29      0.39        38

    accuracy                           1.00     28373
   macro avg       0.79      0.64      0.69     28373
weighted avg       1.00      1.00      1.00     28373
```

XGboost is an optimized version of gradient boosting. As we can see it has a better performance and better fscore in our dataset.

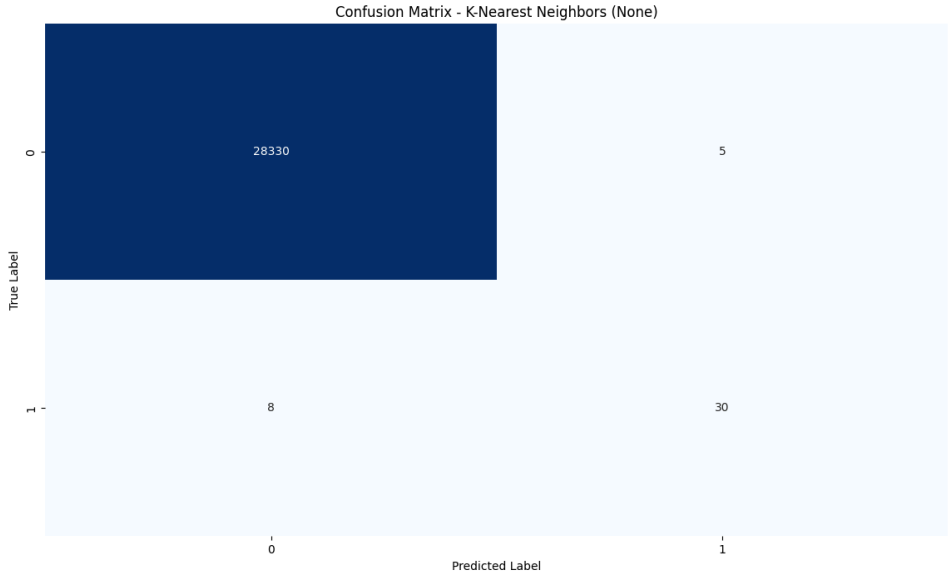
In [30]:
```
model3_xbg = XGBClassifier(n_estimators=100, use_label_encoder=False,
eval_metric="logloss", random_state=13, n_jobs=-1)
model3_xbg_results = evaluate_model("XGBoost", model3_xbg, X_train,
y_train, "None", with_graph=True)
print(model3_xbg_results)
```

```
d:\projects_last\.venv\Lib\site-
packages\xgboost\training.py:199: UserWarning: [22:09:30]
WARNING: C:\actions-
runner\_work\xgboost\xgboost\src\learner.cc:790:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```



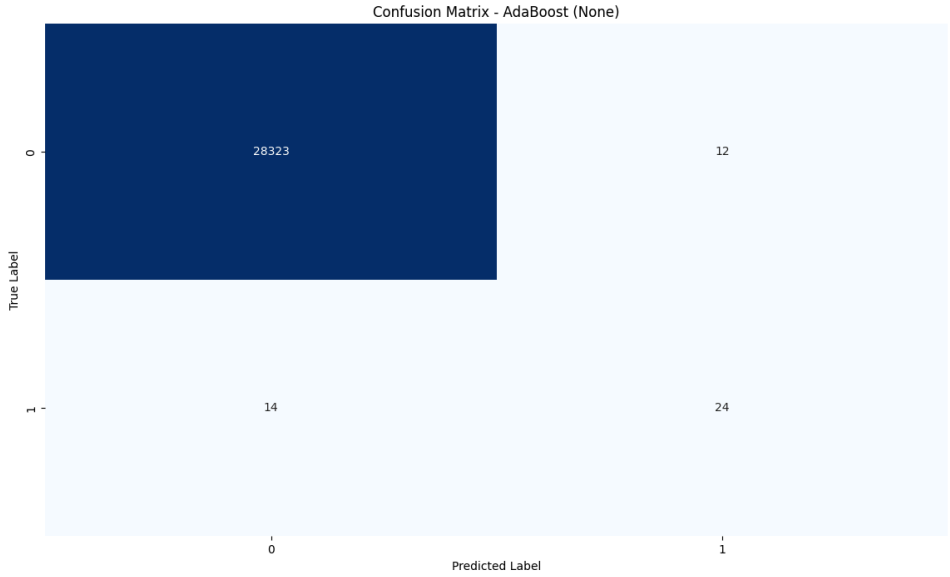
	precision	recall	f1-score	support
0	1.00	1.00	1.00	28335
1	0.78	0.74	0.76	38
accuracy			1.00	28373
macro avg	0.89	0.87	0.88	28373
weighted avg	1.00	1.00	1.00	28373

In [31]:
```
knn = KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
knn_results = evaluate_model("K-Nearest Neighbors", knn, X_train,
y_train, "None", with_graph=True)
print(knn_results)
```



	precision	recall	f1-score	support
0	1.00	1.00	1.00	28335
1	0.86	0.79	0.82	38
accuracy			1.00	28373
macro avg	0.93	0.89	0.91	28373
weighted avg	1.00	1.00	1.00	28373

In [32]:
```
ada_boost = AdaBoostClassifier(n_estimators=100, random_state=13)
ada_boost_results = evaluate_model("AdaBoost", ada_boost, X_train,
y_train, "None", with_graph=True)
print(ada_boost_results)
```



	precision	recall	f1-score	support
0	1.00	1.00	1.00	28335
1	0.67	0.63	0.65	38
accuracy			1.00	28373
macro avg	0.83	0.82	0.82	28373
weighted avg	1.00	1.00	1.00	28373

In [33]:
```
for _ in [0.001, 0.01, 0.1, 0.2, 1, 10, 100]:
    ada_boost = AdaBoostClassifier(n_estimators=100, random_state=13,
learning_rate=_)
    ada_boost_results = evaluate_model("AdaBoost", ada_boost, X_train,
y_train, f"Learning Rate: {_}", with_graph=False)
    print(ada_boost_results)
```

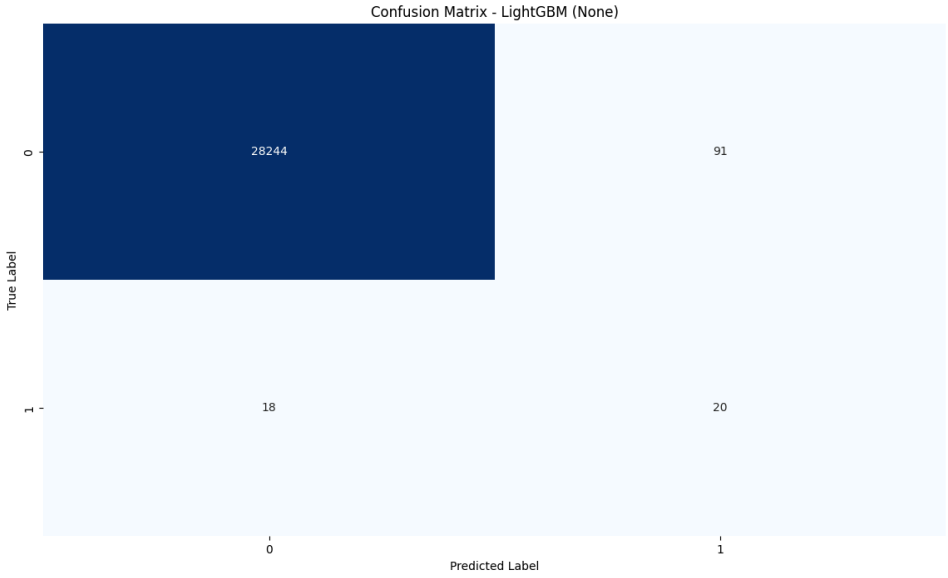
```
                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.63      0.58      0.60        38

        accuracy                           1.00     28373
       macro avg       0.81      0.79      0.80     28373
    weighted avg       1.00      1.00      1.00     28373

                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.73      0.50      0.59        38

        accuracy                           1.00     28373
       macro avg       0.87      0.75      0.80     28373
    weighted avg       1.00      1.00      1.00     28373

                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.75      0.39      0.52        38

        accuracy                           1.00     28373
       macro avg       0.87      0.70      0.76     28373
    weighted avg       1.00      1.00      1.00     28373

                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.83      0.66      0.74        38

        accuracy                           1.00     28373
       macro avg       0.92      0.83      0.87     28373
    weighted avg       1.00      1.00      1.00     28373

                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.67      0.63      0.65        38

        accuracy                           1.00     28373
       macro avg       0.83      0.82      0.82     28373
    weighted avg       1.00      1.00      1.00     28373

                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.63      0.58      0.60        38
```

```
    accuracy                           1.00     28373
   macro avg       0.81      0.79      0.80     28373
weighted avg       1.00      1.00      1.00     28373

              precision    recall  f1-score   support

           0       0.38      0.00      0.00     28335
           1       0.00      0.39      0.00        38

    accuracy                           0.00     28373
   macro avg       0.19      0.20      0.00     28373
weighted avg       0.38      0.00      0.00     28373
```

```
d:\projects_last\.venv\Lib\site-
packages\sklearn\ensemble\_weight_boosting.py:565:
RuntimeWarning: overflow encountered in exp
  sample_weight = np.exp(
d:\projects_last\.venv\Lib\site-packages\sklearn\base.py:1336:
UserWarning: Sample weights have reached infinite values, at
iteration 1, causing overflow. Iterations stopped. Try lowering
the learning rate.
  return fit_method(estimator, *args, **kwargs)
```

In [34]:
```
lgmb = LGBMClassifier(n_estimators=100, random_state=13, n_jobs=-1)
lgmb_results = evaluate_model("LightGBM", lgmb, X_train, y_train,
"None", with_graph=True)
print(lgmb_results)
```

```
[LightGBM] [Info] Number of positive: 388, number of negative:
226592
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the
overhead of testing was 0.025474 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7929
[LightGBM] [Info] Number of data points in the train set: 226980,
number of used features: 32
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.001709 ->
initscore=-6.369901
[LightGBM] [Info] Start training from score -6.369901
```



	precision	recall	f1-score	support
0	1.00	1.00	1.00	28335
1	0.18	0.53	0.27	38
accuracy			1.00	28373
macro avg	0.59	0.76	0.63	28373
weighted avg	1.00	1.00	1.00	28373

In [35]:
```
for _ in [0.001, 0.01, 0.1, 0.2, 1, 10, 100]:
    lgm_model = LGBMClassifier(n_estimators=100, random_state=13,
learning_rate=_)
    lgm_model_results = evaluate_model("LightGBM", lgm_model, X_train,
y_train, f"Learning Rate: {_}", with_graph=False)
    print(lgm_model_results)
```

```
[LightGBM] [Info] Number of positive: 388, number of negative:
226592
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the
overhead of testing was 0.025959 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7929
[LightGBM] [Info] Number of data points in the train set: 226980,
number of used features: 32
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.001709 ->
initscore=-6.369901
[LightGBM] [Info] Start training from score -6.369901
```

```
d:\projects_last\.venv\Lib\site-
packages\sklearn\metrics\_classification.py:1833:
UndefinedMetricWarning: Precision is ill-defined and being set
to 0.0 due to no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
result.shape[0])
d:\projects_last\.venv\Lib\site-
packages\sklearn\metrics\_classification.py:1833:
UndefinedMetricWarning: Precision is ill-defined and being set
to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
result.shape[0])
d:\projects_last\.venv\Lib\site-
packages\sklearn\metrics\_classification.py:1833:
UndefinedMetricWarning: Precision is ill-defined and being set
to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
result.shape[0])
d:\projects_last\.venv\Lib\site-
packages\sklearn\metrics\_classification.py:1833:
UndefinedMetricWarning: Precision is ill-defined and being set
to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
result.shape[0])
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.00      0.00      0.00        38

    accuracy                           1.00     28373
   macro avg       0.50      0.50      0.50     28373
```

```
weighted avg       1.00      1.00      1.00     28373

[LightGBM] [Info] Number of positive: 388, number of negative:
226592
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.007068 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 7929
[LightGBM] [Info] Number of data points in the train set: 226980,
number of used features: 32
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.001709 ->
initscore=-6.369901
[LightGBM] [Info] Start training from score -6.369901
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.91      0.76      0.83        38

    accuracy                           1.00     28373
   macro avg       0.95      0.88      0.91     28373
weighted avg       1.00      1.00      1.00     28373

[LightGBM] [Info] Number of positive: 388, number of negative:
226592
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the
overhead of testing was 0.029836 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7929
[LightGBM] [Info] Number of data points in the train set: 226980,
number of used features: 32
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.001709 ->
initscore=-6.369901
[LightGBM] [Info] Start training from score -6.369901
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.18      0.53      0.27        38

    accuracy                           1.00     28373
   macro avg       0.59      0.76      0.63     28373
weighted avg       1.00      1.00      1.00     28373

[LightGBM] [Info] Number of positive: 388, number of negative:
226592
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the
overhead of testing was 0.038739 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7929
[LightGBM] [Info] Number of data points in the train set: 226980,
number of used features: 32
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.001709 ->
initscore=-6.369901
[LightGBM] [Info] Start training from score -6.369901
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.27      0.37      0.31        38

    accuracy                           1.00     28373
   macro avg       0.64      0.68      0.66     28373
weighted avg       1.00      1.00      1.00     28373

[LightGBM] [Info] Number of positive: 388, number of negative:
226592
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the
overhead of testing was 0.036745 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7929
[LightGBM] [Info] Number of data points in the train set: 226980,
number of used features: 32
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.001709 ->
initscore=-6.369901
[LightGBM] [Info] Start training from score -6.369901
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
```

```
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
```

```
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
              precision    recall  f1-score   support

           0       1.00      0.97      0.98     28335
           1       0.01      0.26      0.02        38

    accuracy                           0.97     28373
   macro avg       0.51      0.62      0.50     28373
weighted avg       1.00      0.97      0.98     28373

[LightGBM] [Info] Number of positive: 388, number of negative: 226592
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.028394 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7929
[LightGBM] [Info] Number of data points in the train set: 226980, number of used features: 32
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.001709 -> initscore=-6.369901
[LightGBM] [Info] Start training from score -6.369901
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
```

```
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
```

```
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
              precision    recall  f1-score   support

           0       1.00      0.87      0.93     28335
           1       0.00      0.00      0.00        38
```

```
    accuracy                           0.87     28373
   macro avg       0.50      0.44      0.47     28373
weighted avg       1.00      0.87      0.93     28373

[LightGBM] [Info] Number of positive: 388, number of negative: 
226592
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the 
overhead of testing was 0.030914 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7929
[LightGBM] [Info] Number of data points in the train set: 226980, 
number of used features: 32
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.001709 -> 
initscore=-6.369901
[LightGBM] [Info] Start training from score -6.369901
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more 
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best 
gain: -inf
```

```
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
```

```
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Stopped training because there are no more leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

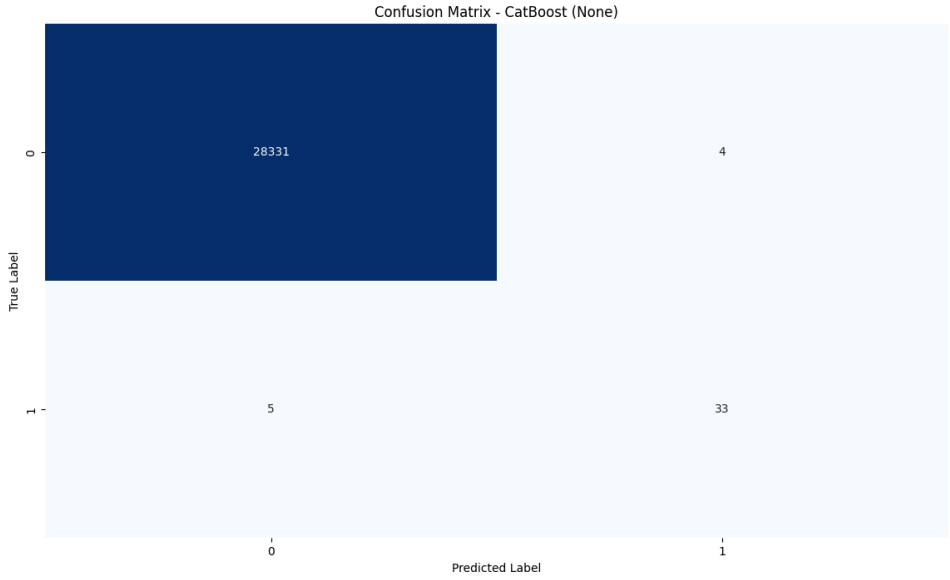
```
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
```

```
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] Stopped training because there are no more
leaves that meet the split requirements
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.44      0.95      0.60        38

    accuracy                           1.00     28373
   macro avg       0.72      0.97      0.80     28373
weighted avg       1.00      1.00      1.00     28373
```

Catboost is a really fast gradiant boosting algorithm that uses categorical boosting.

In [36]:
```
cb_plain = CatBoostClassifier(iterations=200, random_seed=13, verbose=0,
thread_count=-1)
cb_plain_results = evaluate_model("CatBoost", cb_plain, X_train,
y_train, "None", with_graph=True)
print(cb_plain_results)
```



	precision	recall	f1-score	support
0	1.00	1.00	1.00	28335
1	0.89	0.87	0.88	38
accuracy			1.00	28373
macro avg	0.95	0.93	0.94	28373
weighted avg	1.00	1.00	1.00	28373

In [37]:
```
for _ in [0.1, 1, 10, 50, 100]:
    model_catboost = CatBoostClassifier(iterations=200, random_seed=13,
verbose=0, thread_count=-1, class_weights=[1, _])
    res = evaluate_model("CatBoost", model_catboost, X_train, y_train,
f"Class Weights{{0:1, 1:{_}}}")
    print(res)
```

```
                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.97      0.87      0.92        38

        accuracy                           1.00     28373
       macro avg       0.99      0.93      0.96     28373
    weighted avg       1.00      1.00      1.00     28373

                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.89      0.87      0.88        38

        accuracy                           1.00     28373
       macro avg       0.95      0.93      0.94     28373
    weighted avg       1.00      1.00      1.00     28373

                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.95      0.92      0.93        38

        accuracy                           1.00     28373
       macro avg       0.97      0.96      0.97     28373
    weighted avg       1.00      1.00      1.00     28373

                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.85      0.92      0.89        38

        accuracy                           1.00     28373
       macro avg       0.93      0.96      0.94     28373
    weighted avg       1.00      1.00      1.00     28373

                   precision    recall  f1-score   support

               0       1.00      1.00      1.00     28335
               1       0.82      0.87      0.85        38

        accuracy                           1.00     28373
       macro avg       0.91      0.93      0.92     28373
    weighted avg       1.00      1.00      1.00     28373
```

In [41]:
```
results_df = pd.DataFrame(results)
print(f"\n--- FINAL COMPARISON TABLE --- // {results_df.shape}")
print(results_df.sort_values(by="AUPRC",
ascending=False).to_markdown(index=False))
```

```
--- FINAL COMPARISON TABLE --- // (57, 6)
| Model               | Sampling Technique                      |
AUPRC |   Recall |   Precision |   F1 Score |
|:--------------------|:----------------------------------------
|------------:|---------:|------------:|-----------:|
| CatBoost            | Class Weights{0:1, 1:50}                |
0.952778    | 0.921053 | 0.853659    | 0.886076   |
| LightGBM            | Learning Rate: 0.001                    |
0.941123    | 0        | 0           | 0          |
| CatBoost            | Class Weights{0:1, 1:0.1}               |
0.94048     | 0.868421 | 0.970588    | 0.916667   |
| CatBoost            | Class Weights{0:1, 1:1}                 |
0.936896    | 0.868421 | 0.891892    | 0.88       |
| CatBoost            | None                                    |
0.936896    | 0.868421 | 0.891892    | 0.88       |
| LightGBM            | Learning Rate: 0.01                     |
0.934952    | 0.763158 | 0.90625     | 0.828571   |
| CatBoost            | Class Weights{0:1, 1:10}                |
0.904727    | 0.921053 | 0.945946    | 0.933333   |
| CatBoost            | Class Weights{0:1, 1:100}               |
0.881736    | 0.868421 | 0.825       | 0.846154   |
| Neural Network      | 2 Hidden Layers (64, 32)                |
0.854543    | 0.921053 | 0.406977    | 0.564516   |
| K-Nearest Neighbors | None                                    |
0.834235    | 0.789474 | 0.857143    | 0.821918   |
| XGBoost             | None                                    |
0.775012    | 0.736842 | 0.777778    | 0.756757   |
| AdaBoost            | None                                    |
0.767627    | 0.631579 | 0.666667    | 0.648649   |
| AdaBoost            | Learning Rate: 1                        |
0.767627    | 0.631579 | 0.666667    | 0.648649   |
| Logistic Regression | SMOTE'd/Class Weights{0:1, 1:6}         |
0.764148    | 1        | 0.0128596   | 0.0253926  |
| Random forest       | None                                    |
0.761621    | 0.5      | 0.791667    | 0.612903   |
| Random forest       | Class Weights {0:1, 1:1}                |
0.761621    | 0.5      | 0.791667    | 0.612903   |
| AdaBoost            | Learning Rate: 0.2                      |
0.757391    | 0.657895 | 0.833333    | 0.735294   |
| Random forest       | Class Weights {0:1, 1:0.1}              |
0.75059     | 0.289474 | 1           | 0.44898    |
| Logistic Regression | SMOTE'd/Class Weights{0:1, 1:10}        |
0.748426    | 1        | 0.00944099  | 0.0187054  |
| Logistic Regression | SMOTE'd/Class Weights{0:1, 1:1}         |
0.745526    | 1        | 0.0485934   | 0.0926829  |
| Random forest       | Class Weights {0:1, 1:100}              |
0.740873    | 0.947368 | 0.765957    | 0.847059   |
| Logistic Regression | Class Weights{0:1, 1:1000}              |
0.735244    | 1        | 0.0279001   | 0.0542857  |
| Logistic Regression | SMOTE'd/Class Weights{0:1, 1:0.1}       |
```

```
0.734894    | 0.973684 | 0.37        | 0.536232   |
| Logistic Regression | Class Weights{0:1, 1:10}                 |
0.732477    | 0.894737 | 0.708333    | 0.790698   |
| Logistic Regression | Class Weights{0:1, 1:6}                  |
0.731811    | 0.894737 | 0.723404    | 0.8        |
| AdaBoost            | Learning Rate: 0.1                       |
0.726751    | 0.394737 | 0.75        | 0.517241   |
| Logistic Regression | Class Weights{0:1, 1:1}                  |
0.726237    | 0.526316 | 0.740741    | 0.615385   |
| Logistic Regression | None                                     |
0.726237    | 0.526316 | 0.740741    | 0.615385   |
| Logistic Regression | Class Weights{0:1, 1:100}                |
0.71678     | 0.973684 | 0.234177    | 0.377551   |
| Random forest       | Class Weights {0:1, 1:10}                |
0.713888    | 0.789474 | 0.810811    | 0.8        |
| Random forest       | Class Weights {0:1, 1:50}                |
0.68849     | 0.894737 | 0.772727    | 0.829268   |
| Random forest       | SMOTE'd/Class Weights {0:1, 1:0.1}       |
0.67529     | 0.947368 | 0.765957    | 0.847059   |
| Random forest       | Class Weights {0:1, 1:1000}              |
0.674814    | 0.973684 | 0.104816    | 0.189258   |
| AdaBoost            | Learning Rate: 0.01                      |
0.672964    | 0.5      | 0.730769    | 0.59375    |
| Random forest       | SMOTE'd/Class Weights {0:1, 1:1}         |
0.671631    | 0.973684 | 0.20904     | 0.344186   |
| Random forest       | SMOTE'd/Class Weights {0:1, 1:10}        |
0.649826    | 1        | 0.00461445  | 0.00918651 |
| Logistic Regression | SMOTE'd/Class Weights{0:1, 1:100}        |
0.631268    | 1        | 0.00422457  | 0.00841359 |
| Random forest       | SMOTE'd/Class Weights {0:1, 1:100}       |
0.60422     | 1        | 0.00147247  | 0.00294061 |
| AdaBoost            | Learning Rate: 0.001                     |
0.594758    | 0.578947 | 0.628571    | 0.60274    |
| Logistic Regression | Under-sampled/Class Weights{0:1, 1:6}    |
0.586038    | 1        | 0.00931144  | 0.0184511  |
| Logistic Regression | Under-sampled/Class Weights{0:1, 1:10}   |
0.565364    | 1        | 0.00728248  | 0.0144597  |
| Logistic Regression | Under-sampled/Class Weights{0:1, 1:1}    |
0.559581    | 1        | 0.0238245   | 0.0465401  |
| Logistic Regression | Class Weights{0:1, 1:0.1}                |
0.550954    | 0.184211 | 0.777778    | 0.297872   |
| Random forest       | SMOTE'd/Class Weights {0:1, 1:1000}      |
0.517789    | 1        | 0.00142014  | 0.00283624 |
| Logistic Regression | Under-sampled/Class Weights{0:1, 1:0.1}  |
0.439053    | 0.973684 | 0.0833333   | 0.153527   |
| LightGBM            | Learning Rate: 100                       |
0.415988    | 0.947368 | 0.439024    | 0.6        |
| Logistic Regression | Under-sampled/Class Weights{0:1, 1:100}  |
0.414393    | 1        | 0.00396618  | 0.00790103 |
| Logistic Regression | SMOTE'd/Class Weights{0:1, 1:1000}       |
0.413121    | 1        | 0.0026916   | 0.00536875 |
| AdaBoost            | Learning Rate: 10                        |
0.36446     | 0.578947 | 0.628571    | 0.60274    |
| Logistic Regression | Under-sampled/Class Weights{0:1, 1:1000} |
0.316432    | 1        | 0.00267681  | 0.00533933 |
| Gradient Boosting   | None                                     |
0.297955    | 0.289474 | 0.578947    | 0.385965   |
```

```
| LightGBM            | Learning Rate: 0.1                      |
0.121617    | 0.526316 | 0.18018     | 0.268456   |
| LightGBM            | None                                    |
0.121617    | 0.526316 | 0.18018     | 0.268456   |
| LightGBM            | Learning Rate: 0.2                      |
0.101981    | 0.368421 | 0.27451     | 0.314607   |
| LightGBM            | Learning Rate: 1                        |
0.00400818  | 0.263158 | 0.0114679   | 0.021978   |
| LightGBM            | Learning Rate: 10                       |
0.00130512  | 0        | 0           | 0          |
| AdaBoost            | Learning Rate: 100                      |
0.000999237 | 0.394737 | 0.000529362 | 0.00105731 |
```

In [ ]:

```
def objective(trial, model_name, X_tr, y_tr):
    if model_name == "LogisticRegression":
        C = trial.suggest_float("C", 1e-4, 1e2, log=True)
        class_weight = trial.suggest_categorical("class_weight", [None,
"balanced", {0: 1, 1: 10}, {0: 1, 1: 50}])
        model = LogisticRegression(C=C, class_weight=class_weight,
max_iter=1000, random_state=13)

    elif model_name == "RandomForest":
        n_estimators = trial.suggest_int("n_estimators", 50, 200)
        max_depth = trial.suggest_int("max_depth", 3, 15)
        min_samples_split = trial.suggest_int("min_samples_split", 2,
20)
        class_weight = trial.suggest_categorical("class_weight", [None,
"balanced", {0: 1, 1: 10}, {0: 1, 1: 50}])
        model = RandomForestClassifier(n_estimators=n_estimators,
max_depth=max_depth,

min_samples_split=min_samples_split, class_weight=class_weight,
                                       random_state=13, n_jobs=-1)

    elif model_name == "XGBoost":
        learning_rate = trial.suggest_float("learning_rate", 0.01, 0.3,
log=True)
        max_depth = trial.suggest_int("max_depth", 3, 10)
        n_estimators = trial.suggest_int("n_estimators", 50, 200)
        scale_pos_weight = trial.suggest_float("scale_pos_weight", 1,
100)
        model = XGBClassifier(learning_rate=learning_rate,
max_depth=max_depth, n_estimators=n_estimators,
                              scale_pos_weight=scale_pos_weight,
eval_metric="logloss", random_state=13, n_jobs=-1)

    elif model_name == "CatBoost":
        learning_rate = trial.suggest_float("learning_rate", 0.01, 0.3,
log=True)
        max_depth = trial.suggest_int("max_depth", 3, 10)
        iterations = trial.suggest_int("iterations", 100, 300)
        class_weights = trial.suggest_categorical("class_weights", [[1,
10], [1, 50], [1, 100]])
        model = CatBoostClassifier(learning_rate=learning_rate,
max_depth=max_depth, iterations=iterations,
                                   class_weights=class_weights,
verbose=0, random_seed=13, thread_count=-1)

    elif model_name == "KNN":
        n_neighbors = trial.suggest_int("n_neighbors", 3, 15)
        weights = trial.suggest_categorical("weights", ["uniform",
"distance"])
        model = KNeighborsClassifier(n_neighbors=n_neighbors,
weights=weights, n_jobs=-1)

    # Train and evaluate
    model.fit(X_tr, y_tr)
    y_prob = model.predict_proba(X_val)[:, 1]
    auprc = average_precision_score(y_val, y_prob)
```

```
    return auprc


def run_hyperparameter_tuning(model_name, n_trials=100):

    sampler = TPESampler(seed=13)
    pruner = MedianPruner(n_warmup_steps=5)

    study = optuna.create_study(direction="maximize", sampler=sampler,
pruner=pruner)
    study.optimize(lambda trial: objective(trial, model_name, X_train,
y_train),
                   n_trials=n_trials, show_progress_bar=True)

    print(f"\n{"="*60}")
    print(f"Best trial for {model_name}:")
    print(f"{"="*60}")
    print(f"AUPRC: {study.best_value:.4f}")
    print(f"Best hyperparameters: {study.best_params}")

    return study.best_params, study


best_params, study = run_hyperparameter_tuning("CatBoost", n_trials=50)
```

```
============================================================
Best trial for CatBoost:
============================================================
AUPRC: 0.9468
Best hyperparameters: {'learning_rate': 0.03810210980374369, 'max_depth': 10,
'iterations': 147, 'class_weights': [1, 10]}
```

I cleared the result of the previous cell because it was kinda ugly with all errors etc.

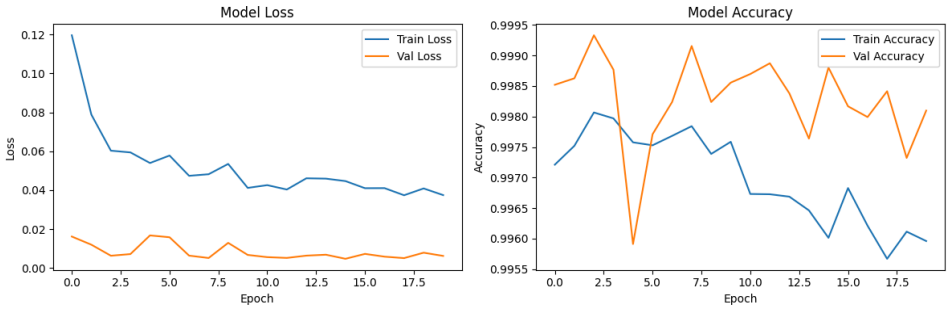
In [ ]:

```
warnings.filterwarnings("ignore")

# Scaling for nn
scaler_nn = StandardScaler()
X_train_scaled = scaler_nn.fit_transform(X_train)
X_val_scaled = scaler_nn.transform(X_val)
X_test_scaled = scaler_nn.transform(X_test)

# Build nn with 2 hidden layers
model_nn = keras.Sequential([
    layers.Input(shape=(X_train_scaled.shape[1],)),
    layers.Dense(64, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(32, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(1, activation="sigmoid")
])

# Compile with class weight adjustment with accuracy metric
model_nn.compile(
    optimizer="adam",
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

# Train the model
history = model_nn.fit(
    X_train_scaled, y_train,
    validation_data=(X_val_scaled, y_val),
    epochs=20,
    batch_size=32,
    class_weight={0: 1, 1: 50},
    verbose=1
)

# Evaluate on validation set
y_pred_nn = model_nn.predict(X_val_scaled)
y_pred_nn_binary = (y_pred_nn > 0.5).astype(int).flatten()

auprc_nn = average_precision_score(y_val, y_pred_nn)
recall_nn = recall_score(y_val, y_pred_nn_binary)
precision_nn = precision_score(y_val, y_pred_nn_binary)
f1_nn = f1_score(y_val, y_pred_nn_binary)

results.append({
    "Model": "Neural Network",
    "Sampling Technique": "2 Hidden Layers (64, 32)",
    "AUPRC": auprc_nn,
    "Recall": recall_nn,
    "Precision": precision_nn,
    "F1 Score": f1_nn
})

print(f"Neural Network Results:")
print(f"AUPRC: {auprc_nn:.4f}, Recall: {recall_nn:.4f}, Precision:
{precision_nn:.4f}, F1: {f1_nn:.4f}")
print(classification_report(y_val, y_pred_nn_binary))
```

```
# Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history["loss"], label="Train Loss")
plt.plot(history.history["val_loss"], label="Val Loss")
plt.title("Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history["accuracy"], label="Train Accuracy")
plt.plot(history.history["val_accuracy"], label="Val Accuracy")
plt.title("Model Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.tight_layout()
plt.show()
```

```
Epoch 1/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m12s[0m 2ms/step - accuracy: 0.9972 - loss: 0.1197 -
val_accuracy: 0.9985 - val_loss: 0.0161
Epoch 2/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9975 - loss: 0.0789 -
val_accuracy: 0.9986 - val_loss: 0.0119
Epoch 3/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 1ms/step - accuracy: 0.9981 - loss: 0.0603 -
val_accuracy: 0.9993 - val_loss: 0.0063
Epoch 4/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9980 - loss: 0.0594 -
val_accuracy: 0.9988 - val_loss: 0.0071
Epoch 5/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9976 - loss: 0.0540 -
val_accuracy: 0.9959 - val_loss: 0.0167
Epoch 6/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 1ms/step - accuracy: 0.9975 - loss: 0.0578 -
val_accuracy: 0.9977 - val_loss: 0.0158
Epoch 7/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9977 - loss: 0.0474 -
val_accuracy: 0.9982 - val_loss: 0.0063
Epoch 8/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 1ms/step - accuracy: 0.9978 - loss: 0.0482 -
val_accuracy: 0.9992 - val_loss: 0.0051
Epoch 9/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 1ms/step - accuracy: 0.9974 - loss: 0.0535 -
val_accuracy: 0.9982 - val_loss: 0.0129
```

```
Epoch 10/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9976 - loss: 0.0412 -
val_accuracy: 0.9986 - val_loss: 0.0067
Epoch 11/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9967 - loss: 0.0426 -
val_accuracy: 0.9987 - val_loss: 0.0055
Epoch 12/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9967 - loss: 0.0403 -
val_accuracy: 0.9989 - val_loss: 0.0051
Epoch 13/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9967 - loss: 0.0461 -
val_accuracy: 0.9984 - val_loss: 0.0063
Epoch 14/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9965 - loss: 0.0459 -
val_accuracy: 0.9976 - val_loss: 0.0068
Epoch 15/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9960 - loss: 0.0447 -
val_accuracy: 0.9988 - val_loss: 0.0047
Epoch 16/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9968 - loss: 0.0410 -
val_accuracy: 0.9982 - val_loss: 0.0072
Epoch 17/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9962 - loss: 0.0410 -
val_accuracy: 0.9980 - val_loss: 0.0058
Epoch 18/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9957 - loss: 0.0374 -
val_accuracy: 0.9984 - val_loss: 0.0050
Epoch 19/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9961 - loss: 0.0409 -
val_accuracy: 0.9973 - val_loss: 0.0078
Epoch 20/20
[1m7094/7094[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m
[1m11s[0m 2ms/step - accuracy: 0.9960 - loss: 0.0375 -
val_accuracy: 0.9981 - val_loss: 0.0062
[1m887/887[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m1s[0m
695us/step
Neural Network Results:
AUPRC: 0.8545, Recall: 0.9211, Precision: 0.4070, F1: 0.5645
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28335
           1       0.41      0.92      0.56        38

    accuracy                           1.00     28373
   macro avg       0.70      0.96      0.78     28373
weighted avg       1.00      1.00      1.00     28373
```



Neural Network Results:
AUPRC: 0.8545, Recall: 0.9211, Precision: 0.4070, F1: 0.5645

As we can see, if we train our model for accuracy in this kind of imbalanced dataset; our precision will be ruined.

We selected AUPRC (Area Under the Precision-Recall Curve) as our primary metric. AUPRC focuses specifically on the positive class and does not give credit for correctly classifying the massive number of normal transactions.

Best Model: The CatBoost Classifier with Class Weights (1:50) yielded the best performance.

AUPRC: 0.95

Recall: 0.92

Precision: 0.85

Conclusion: While Tree-based Ensemble methods (CatBoost, LightGBM, XGBoost) generally outperformed linear models, CatBoost demonstrated superior handling of the imbalanced data without requiring extensive preprocessing or synthetic sampling (like SMOTE). The Neural Network showed promise but was more computationally expensive for a wrose level of performance. I tried to optimise the nn for AUPRC but how many times i tried, it took so long and sometimes python just fails.