Credit Card Fraud Detection - Case Study

Michał Ciesielski 145325

Opis danych

Pochodzenie danych: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Kradzieże kart kredytowych to wciąż poważny ogólnoświatowy problem, który powoduje znaczące straty dla firm, banków czy zwykłych ludzi. Wraz ze wzrostem płatności dokonywanych cyfrowo, oszuści mają więcej okazji by wykorzystać luki w bezpieczeństwie i wykraść ważne informacje. Dana kradzież nie oznacza jedynie strat materialnych, ale również wizerunkowe dla instytucji finansowych i zmniejsza zaufanie klientów wobec tych firm. Wykrycie i zapobieganie kradzieży kart kredytowych jest bardzo ważne w celu zminimalizowania jej negatywnego wpływu. W tym celu doskonale sprawdzą się algorytmy uczenia maszynowego do zidentyfikowania podejrzanych transakcji.

Wybrany zbiór danych przedstawia ważny problem, z którym codziennie zmagają się firmy zajmujące się wydawaniem kart kredytowych. Posłuży on do predykcji transakcji dokonywanych przy użyciu kradzionych kart kredytowych. Zbudowany w ten sposób model pozwoli na nieobciążanie rachunku klienta, jeśli dana transakcja została zainicjowana przy użyciu kradzionej karty kredytowej.

Zbiór danych zawiera transakcje wykonane przy użyciu kart kredytowych w ciągu 2 dni we wrześniu 2013 roku przez Europejczyków. Łączna liczba transakcji w zbiorze to 284807 transakcje. Ważnym problemem jest fakt, że dane dotyczące wykrycia kradzieży są wysoce-niezbalansowane. Liczba nielegalnych transakcji jest znacznie mniejsza niż liczba poprawnych tranakcji. W takiej sytuacji, klasyczne algorytmy uczenia maszynowego słabo radzą sobie z wykrywaniem przypadków należących do klasy mniejszościowej (nielegalne transakcje). Wyzwanie polega więc na zbudowaniu takich modeli, które będą w stanie dokładnie zidentyfikować podejrzane przypadki płatności. Niezbalansowanie danych stanowi znaczne utrudnienie w osiągnięciu właściwego celu, jako że większość klasyfikatorów jest stworzonych w celu maksymalizacji dokładności, która nie jest właściwą metryką w takich przypadkach.

Eksploracyjna analiza danych

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as mpatches
```

```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from imblearn.metrics import geometric_mean_score
from sklearn.metrics import accuracy_score, roc_auc_score, confusion_matrix, recall
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
df = pd.read_csv('creditcard.csv')
# df = df.drop('id', axis=1)
df
```

Out[2]:

	Time	V1	V2	V3	V4	V5	V6	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.2395
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.0788
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.7914
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.2376
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.5929
...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.9182
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.0243
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.2968
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.6861
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.5770

284807 rows × 31 columns

Przykładowy wiersz składa się z 30 atrybutów warunkowych, z czego aż 28 jest wynikiem zastosowania transformacji PCA w ramach preprocessingu. Ze względu na charakter tej metody, nie jestem w stanie uzyskać lepszego kontekstu dla tych atrybutów. Poza tymi atrybutami wyróżniamy również atrybuty 'Time' oraz 'Amount'. Atrybut 'Time' określa czas upływający w sekundach pomiędzy daną transakcją a pierwszą transakcją w zbiorze. Nie ma on większego znaczenia w kontekście badanego problemu (pełni rolę pewnego atrybutu porządkowego). 'Amount' z kolei określa kwotę transakcji, którą będzie obciążony posiadacz karty. Atrybutem decyzyjnym jest flaga, która określa, czy dana transakcja została wykonana przy użyciu karty kupującego (0) czy przy użyciu kradzionej karty (1).

Tabela rozkładu wartości poszczególnych atrybutów

In [3]:
```
df.describe()
```

Out[3]:

	Time	V1	V2	V3	V4	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.8480
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.6040
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.3802
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.1374
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.9159
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.4335
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.1192
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.4801

8 rows × 31 columns

Powyższa tabela pokazuje rozkład wartości, w tym takie charakterystyczne cechy jak np. średnia i odchylenie standardowe, przyjmowane przez poszczególne atrybuty. Jedyną istotną informację niesie ze sobą, poza wspomnianym wcześniej niezbalansowaniem atrybutu decyzyjnego, rozkład atrybutu warunkowego 'Amount'. 75% wartości tego atrybutu osiąga wartości mniejsze niż 77,17. Większość dokonywanych transakcji to transakcje niskobudżetowe, a wśrod nich wiele potencjalnych kradzieży, ze względu na słabsze zabezpieczenia dla takowych operacji.

In [4]:
```
total = df.isna().sum().sort_values(ascending=False)
total.head()
```

Out[4]:
```
Time      0
V16       0
Amount    0
V28       0
V27       0
dtype: int64
```

Analiza brakujących wartości - brak takowych w badanym zbiorze.

Wykresy pudełkowe zawierające kwotę i klasę decyzyjną

In [6]:
```
colors = ['#36ada4', '#f77189']

fig, ax = plt.subplots(ncols=2, figsize=(10,4))


sns.boxplot(data=df,
            x="Class",
            y="Amount",
            hue="Class",
            showfliers=True,
            ax=ax[0])
```

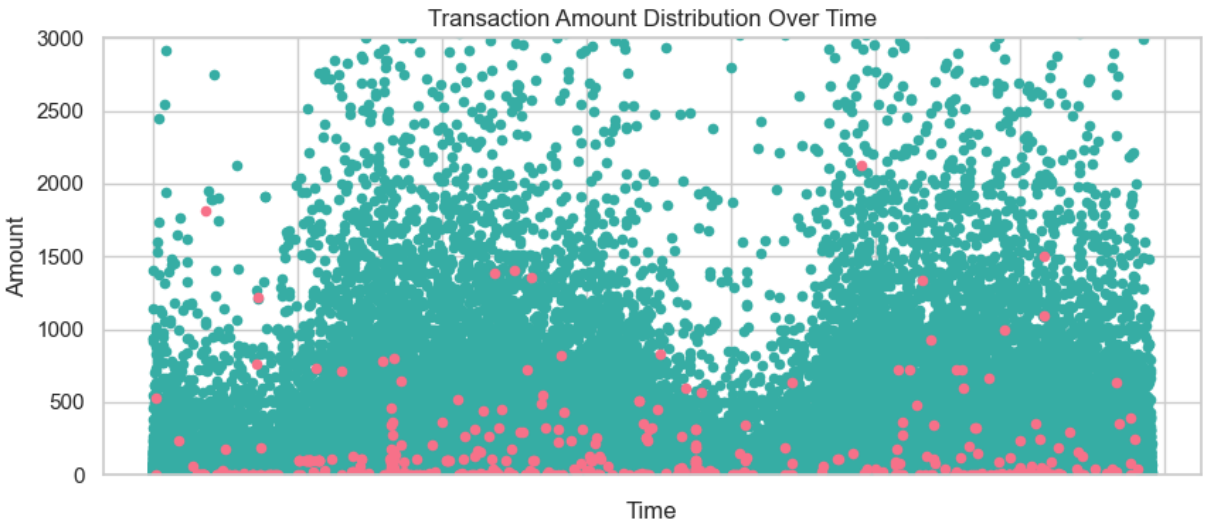
```
sns.boxplot(data=df,
            x="Class",
            y="Amount",
            hue="Class",
            palette=colors,
            showfliers=False,
            ax=ax[1])
sns.set_theme(
    style='whitegrid',
    palette=sns.color_palette(colors)
)

# Add titles to the plots
ax[0].set_title("Transaction Amount Box Plot (Including Fliers)")
ax[1].set_title("Transaction Amount Box Plot (Excluding Fliers)")

# Update legend labels
legend_labels = ['Non-fraud', 'Fraud']
for i in range(2):
    handles, _ = ax[i].get_legend_handles_labels()
    ax[i].legend(handles, legend_labels)


plt.show()
```

Wnioski:

- nielegalne transakcje posiadają niższą wartość mediany
- nielegalne transakcje mają również większy rozstęp międzykwartylowy
- nie istnieją transakcje kradzionymi kartami o kwotach powyżej 3000 więc jest mało prawdopodobne aby duże transakcje odbywały się kradzionymi kartami (np. ze względu na limity płatności kartą bez znajomości kodu PIN).

Wykres punktowy kwoty od czasu

In [7]:
```
f, ax = plt.subplots(figsize=(10, 4))

sns.scatterplot(data=df.loc[df.Class==0],
                x='Time',
                y='Amount',
                color='#36ada4',
                s=30,
                alpha=1,
                linewidth=0)

ax.set(xlabel=None, xticklabels=[])
plt.ylim(0, 3000)


sns.scatterplot(data=df.loc[df.Class==1],
                x='Time',
                y='Amount',
                color='#f77189',
                s=30,
                alpha=1,
                linewidth=0)
plt.ylim(0, 3000)

# Add title to the plot
ax.set_title("Transaction Amount Distribution Over Time")

plt.show()
```

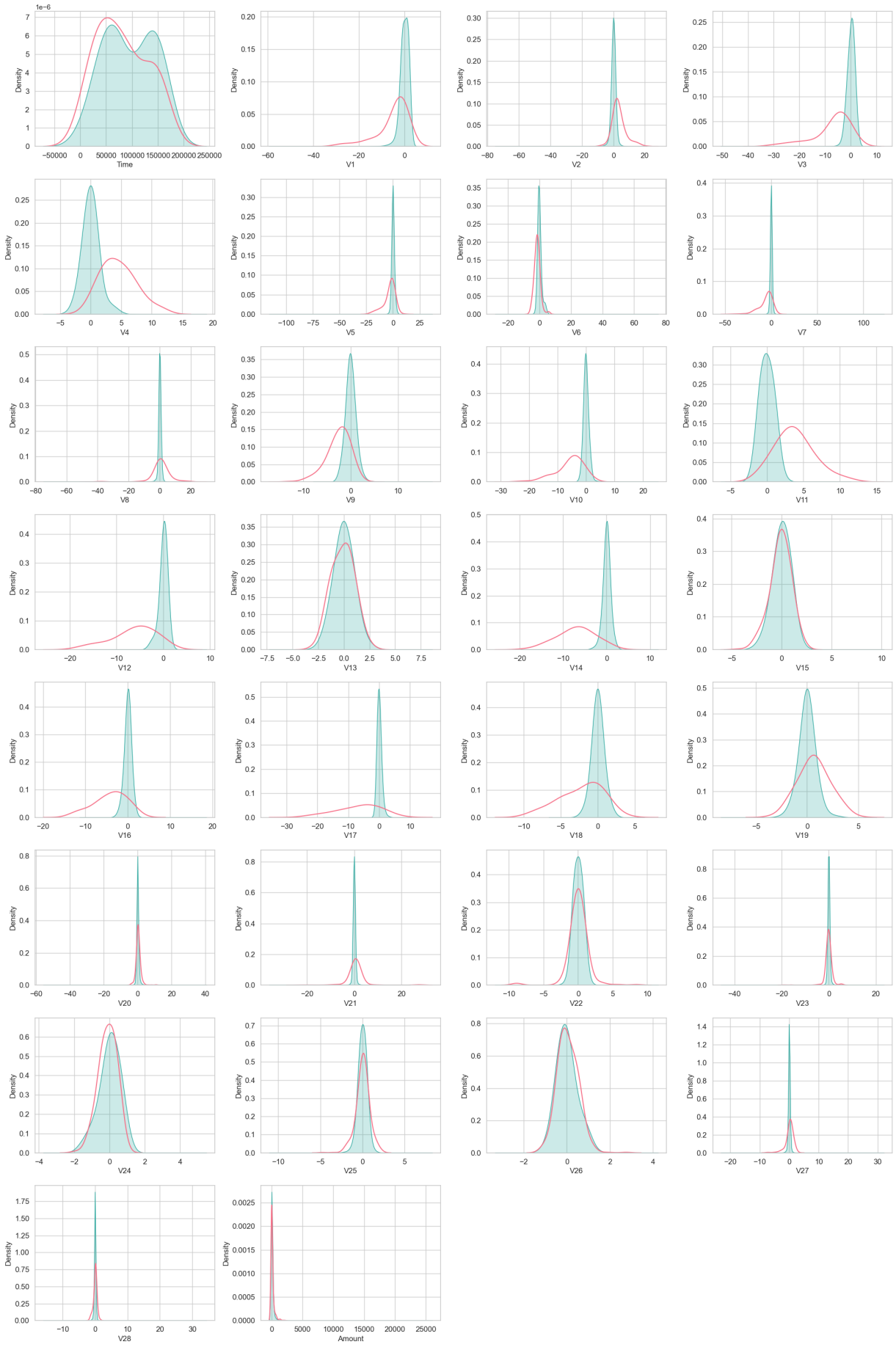


Wnioski:

- jak można zauważyć na wykresie kwoty od czasu dokonania transakcji, nielegalne transakcje występują równomiernie pomiędzy legalnymi transakcjami - brak wyraźnego związku

Wykresy KDE porównujące rozkłady wszystkich atrybutów dla każdej klasy

In [8]:
```
var = df.columns.values

t0 = df.loc[df['Class'] == 0]
t1 = df.loc[df['Class'] == 1]

num_features = len(var)
num_rows = num_features // 4 + int(num_features % 4 != 0)

fig, ax = plt.subplots(nrows=num_rows, ncols=4, figsize=(20, 30))

for idx, feature in enumerate(var):
    row = idx // 4
    col = idx % 4

    sns.kdeplot(t0[feature], bw_method=0.5, label="Class = 0", color='#36ada4', fil
    sns.kdeplot(t1[feature], bw_method=0.5, label="Class = 1", color='#f77189', war

    ax[row, col].set_xlabel(feature, fontsize=12)
    ax[row, col].tick_params(axis='both', which='major', labelsize=12)

# Remove any unused subplots + class
for remaining_ax in ax.flatten()[num_features-1:]:
    remaining_ax.remove()

plt.tight_layout()
plt.show()
```


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

Wnioski:

- Dla niektórych z atrybutów rozkłady są niemal identyczne między legalnymi a nielegalnymi transakcjami
- Inne atrybuty mają wyraźne różnice w średniej i spłaszczeniu, co może być użyteczne w predykcjach.

Ze względu na różne rozkłady poszczególnych atrybutów, każdy z nich zostanie poddany standaryzacji w procesie budowania modelu. Można zauważyć liczne obserwacje odstające dla atrybutu 'Amount', zwłaszcza dla grupy zaklasyfikowanej jako transakcje legalne.

In [9]:
```
display(pd.DataFrame(df['Class'].value_counts()).reset_index())
display(pd.DataFrame(df['Class'].value_counts(normalize=True) * 100).reset_index())
```

	Class	count
0	0	284315
1	1	492

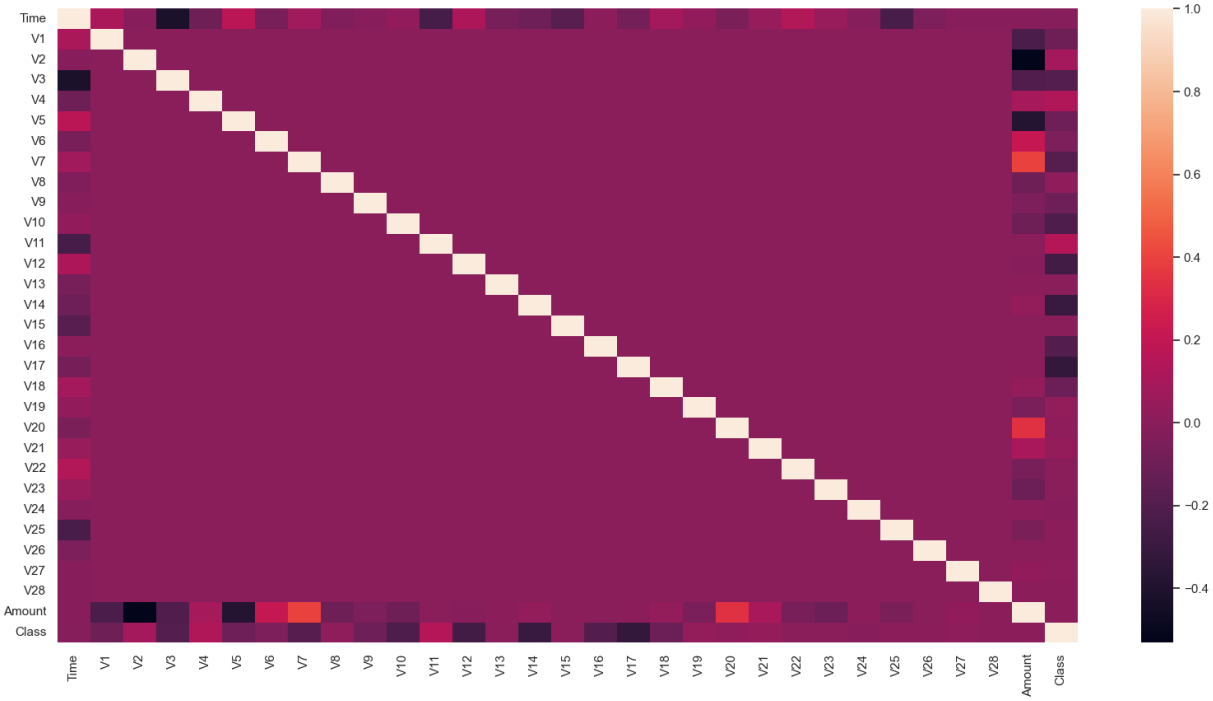
	Class	proportion
0	0	99.827251
1	1	0.172749

Jak można zauważyć, dane są mocno niezbalansowane. Tylko ok. 0,17% przypadków zostało sklasyfikowanych jako transakcje skradzioną kartą w podanym zbiorze.

Macierz korelacji

In [21]:
```
plt.figure(figsize=(20, 10))
sns.heatmap(df.corr())
```

Out[21]: <Axes: >



Wnioski:

- Atrybuty PCA nie są skorelowane ze sobą
- Większość atrybutów PCA słabo koreluje z czasem transakcji czy też kwotą

Budowa modelu

Do budowy końcowego modelu posłużę się trzema dostępnymi klasyfikatorami:

1. LogisticRegression
2. RandomForestClassifier
3. GaussianNB

Wykonanie predykcji posługując się powyższymi algorytmami będzie prostym przepływem, poprzedzonym standaryzacją danych. Z uwagi na wysoki stopień niezbalansowania danych, konieczne będzie zastosowanie technik poprawiających zdolności predykcyjne poszczególnych modeli. Dla metod posiadających parametr *class_weight*, ustawienie jego wartości na 'balanced' poprawi jakość predykcji. Z kolei dla metody GaussianNB w tym celu wykorzystam oversampling przy wykorzystaniu metody SMOTE. Metrykami służacymi do oceny trafności poszczególnych modeli będą tradycyjna trafność, G-mean oraz ROC AUC. Wyznaczając końcowy model skorzystam z podejścia ensemble learning, służącego do połączenia testowanych modeli oraz stworzenia jednego optymalnego modelu predykcyjnego.

Wykorzystane metryki:

- trafność (accuracy) - liczba poprawnych / liczba wszystkich predykcji. Siłą tej metryki będzie niski stopień transakcji odrzuconych. Z drugiej strony ignoruje ona prawdziwie nielegalne transkacje. Z tego powodu stanowi dobry benchmark, jednak nie jest właściwą metryką dla specyfiki danych niezbalansowanych.
- precyzja (precision) - liczba odgadniętych poprawnie / łączna liczba wszystkich predykcji. Określa jaka część wyników wskazanych przez klasyfikator jako dodatnie (transakcje nielegalne) jest faktycznie dodatnia. Jej siła polega na właściwej identyfikacji nielegalnych transakcji.
- czułość (recall) - liczba wyłapanych nielegalnych transakcji / łączna liczba nielegalnych transakcji. Określa jaką część dodatnich wyników wykrył klasyfikator.
- średnia geometryczna (G-mean) - metryka, która maksymalizuje dokładność dla każdej klasy, trzymając te dokładności zbalansowane. Dla klasyfikacji binarnej G-mean jest pierwiastkiem kwadratowym iloczynu czułości (sensitivity, recall) i swoistości (specifity), która określa jaką część ujemnych (niedodatnich) wyników wykrył klasyfikator.
- pole pod krzywą ROC (ROCAUC) - krzywe ROC to wizualizacja zależności pomiędzy skutecznością klasyfikacji pozytywnych (czułość) a nieskutecznością klasyfikacji przypadków negatywnych (1 - swoistość). Bardzo popularnym podejściem jest wyliczanie pola pod wykresem krzywej ROC, oznaczanego jako AUC (ang. area under curve), i traktowanie go jako miarę dobroci i trafności danego modelu. Wartość wskaźnika AUC przyjmuje wartości z przedziału [0; 1]; im większa, tym lepszy model.

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(df.drop(columns=["Class", "Time
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- Regresja logistyczna

In [22]:
```
results = []
lr_clf = LogisticRegression(penalty='l2', class_weight='balanced', random_state=42,
lr_clf.fit(X_train, y_train)
y_pred = lr_clf.predict(X_test)
results.append(dict(name='Logistic Regression', acc=accuracy_score(y_test, y_pred),
                    rocauc=roc_auc_score(y_test, y_pred), precision=precision_score
```

- Las losowy

In [23]:
```
rf_clf = RandomForestClassifier(n_estimators=10, class_weight='balanced', random_st
rf_clf.fit(X_train, y_train)
y_pred = rf_clf.predict(X_test)
results.append(dict(name='Random Forest', acc=accuracy_score(y_test, y_pred), gmean
                    rocauc=roc_auc_score(y_test, y_pred), precision=precision_score
```

- Naiwny Bayes

W podejściu dla klasyfikatora GaussianNB nie można wykorzystać dostosowania wag odwrotnie częstotliwości wysepowania danej klasy ze względu na charakter tej metody. W tym celu wykorzystam metodę nadpróbkowania SMOTE. Metoda ta generuje sztuczne przykłady z klasy mniejszościowej w celu odpowiedniego zrównoważenia występowania przykładów z obu klas.

In [24]:
```
oversample = SMOTE(sampling_strategy="minority", random_state=4)
X_train_smote, y_train_smote = oversample.fit_resample(X_train, y_train)
y_train_smote.value_counts()
```

Out[24]:
```
Class
0    213226
1    213226
Name: count, dtype: int64
```

In [25]:
```
nb_clf = GaussianNB()
nb_clf.fit(X_train_smote, y_train_smote)
y_pred = nb_clf.predict(X_test)
results.append(dict(name='Naive Bayes', acc=accuracy_score(y_test, y_pred), gmean=g
                    rocauc=roc_auc_score(y_test, y_pred), precision=precision_score
```

In [26]:
```
results = pd.DataFrame(results)
results
```

Out[26]:

	name	acc	gmean	rocauc	precision	recall
0	Logistic Regression	0.975268	0.947455	0.947854	0.056034	0.920354
1	Random Forest	0.999508	0.862149	0.871639	0.933333	0.743363
2	Naive Bayes	0.976841	0.920501	0.922136	0.056582	0.867257

- Ensemble learning

In [27]:
```
lr_pred = lr_clf.predict_proba(X_test)[:, 1]
rf_pred = rf_clf.predict_proba(X_test)[:, 1]
nb_clf = nb_clf.predict_proba(X_test)[:, 1]

coefficients = [1, 0.1, 0.1]
ensemble_preds = np.sum([coefficients[0] * lr_pred + coefficients[1] * rf_pred + co
```

- Ewaluacja modelu

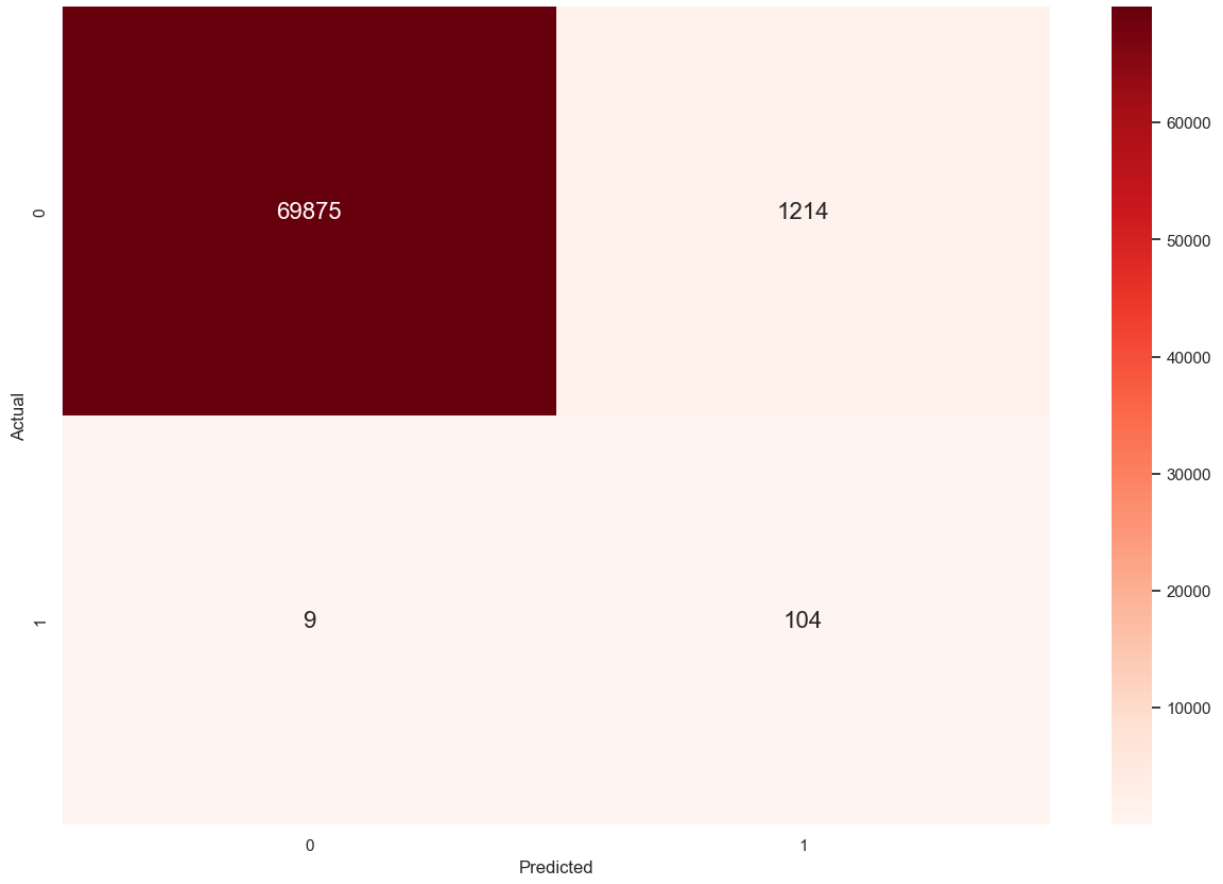
In [28]:
```
results = [dict(name='Ensemble method', acc=accuracy_score(y_test, (ensemble_preds
                rocauc=roc_auc_score(y_test, y_pred), precision=precision_score(y_t
```

In [29]:
```
pd.DataFrame(results)
```

Out[29]:

	name	acc	gmean	rocauc	precision	recall
0	Ensemble method	0.982824	0.951124	0.922136	0.056582	0.867257

In [30]:
```
cm = confusion_matrix(y_test, (ensemble_preds > 0.5))
plt.figure(figsize=(15, 10))
sns.heatmap(cm, annot=True, cmap='Reds', fmt='g', annot_kws={"size": 16})
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



W końcowym modelu uznałem, że regresja logistyczna najlepiej poradziła sobie z problemem niezrównoważenia atrybutu decyzyjnego w podanym zbiorze, dlatego jej współczynnik jest 10-krotnie wyższy. Końcowy model to przykład zastosowania ensemble learning z wykorzystaniem średniej ważonej. Ze względu na fakt, że błąd polegający na wykryciu niestniejącej kradzieży i anulowanie transakcji jest mniej kosztowny niż potencjalny brak wykrycia kradzieży karty, uzyskane wyniki predykcji wydają się satysfakcjonujące. Problem niezbalansowanych danych został więc rozwiązany przy pomocy odpowiedniego zrównoważenia wag dla danej klasy bądź przy zastosowaniu oversamplingu i metody SMOTE. Połączenie metod, korzystając z prostej idei ensemble learning umożliwił stworzenie bardziej ogólnego modelu łączącego mocne strony wcześniej wypracowanych modeli. Skupiając się na powyższej macierzy pomyłek, można odnieść wrażenie, że założony cel został spełniony. Uzyskana niska wartość precyzji, kosztem wysokiej wartości czułości,

pozwala stwierdzić, że zbudowany model nie będzie pomijał nielegalnych transakcji a tym samym kradzieży pieniędzy z kont klientów poszczególnych banków.