



ITADATAhack



Con il patrocinio di

Indice



*“If you torture the data long enough, it will
confess to anything”
Ronald Coase*

- 1 C’era una volta
- 2 Colpo di scena
- 3 Scaviamo più a fondo
- 4 Lampo di genio?
- 5 Non tutto oro quello che luccica
- 6 Addestramento e risultati Task 1
- 7 Addestramento e risultati Task 2

Team



Gabriele Lo Cascio



Gabriele La Milia



Simone Salvatore La Milia

C'ERA UNA VOLTA...

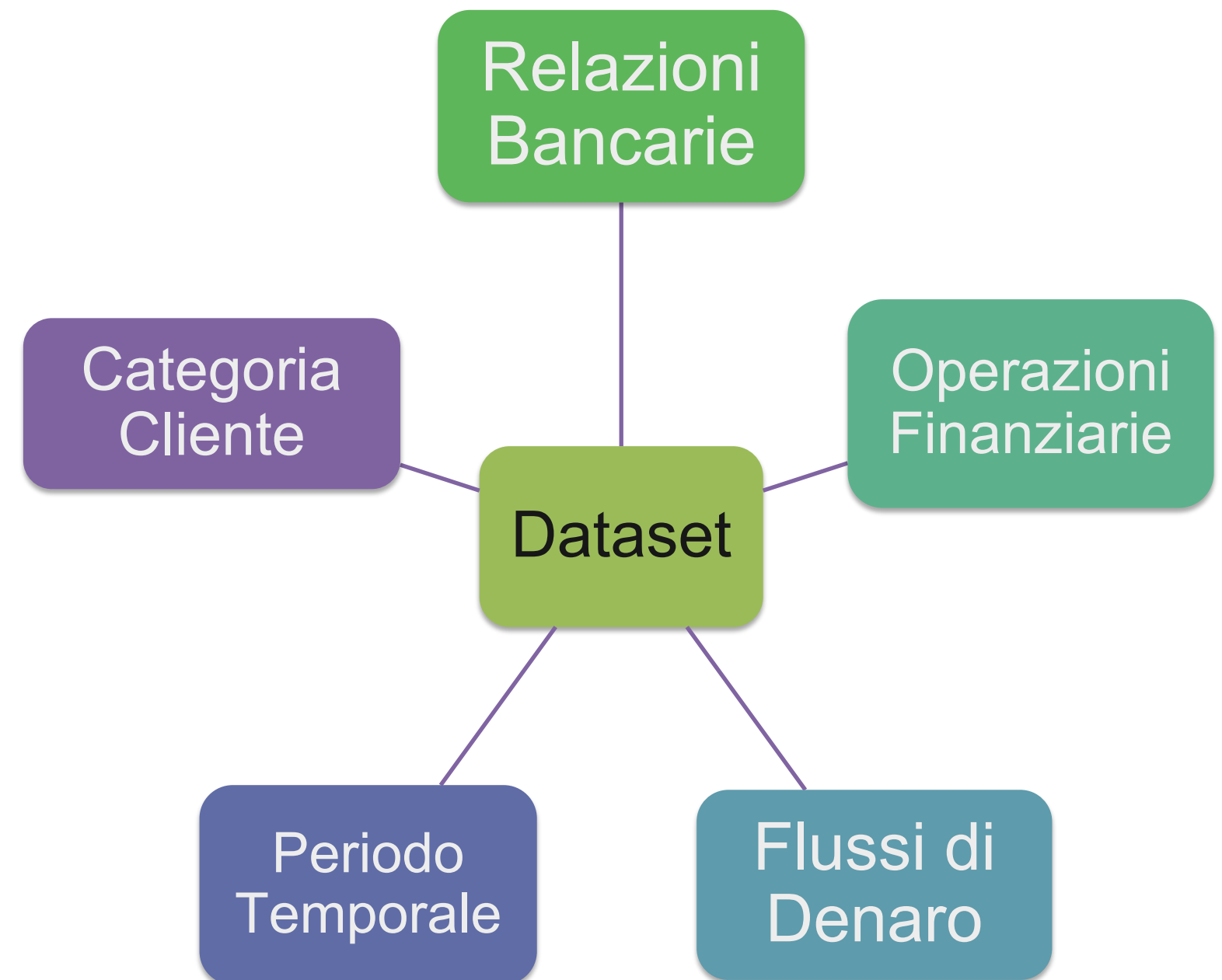
L'obiettivo è costruire un modello predittivo per determinare la probabilità che un cliente ripaghi il suo debito in base alla feature *repays_debt*.

Analizzando le altre features del dataset si è notato che queste possono essere raggruppate in cinque grandi macroaree.

Il dataset contiene informazioni per ogni cliente per un periodo di cinque anni, che vengono suddivisi in venti trimestri. Di fatto la dimensione del dataset si ottiene per costruzione: si moltiplica il numero di clienti unici e i periodi da analizzare. Il numero totale di clienti si ottiene come segue.

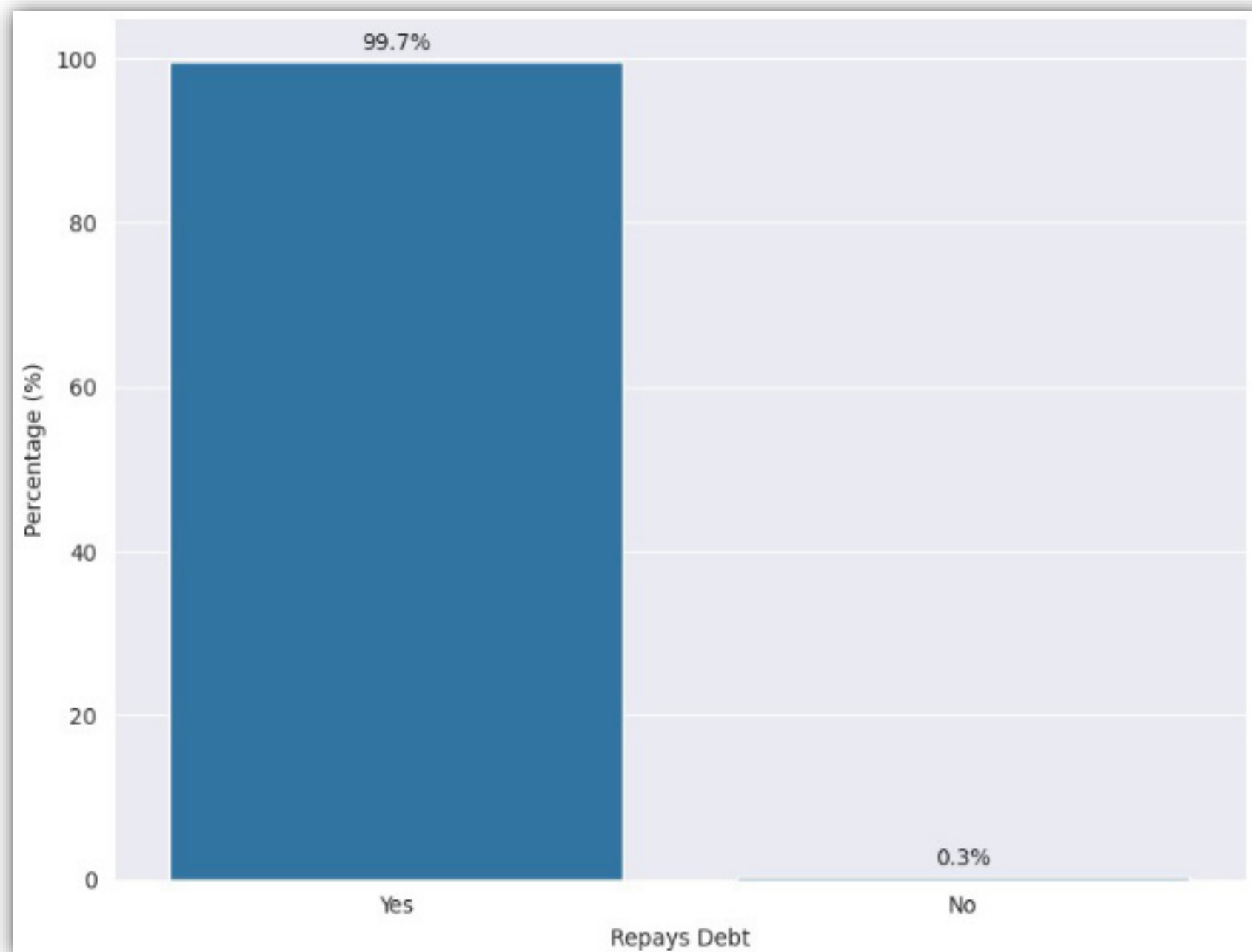
```
print(f'There are {df["client_id"].unique().size} clients')
sns.barplot(x=['Unique Clients'], y=[df['client_id'].nunique()])

There are 102808 clients
```



COLPO DI SCENA

La colonna perno di questa challenge è proprio *repays_debt*. Ci siamo chiesti quindi...
Quanti clienti abbiamo che sono riconosciuti come “buoni”?
E quanti come “cattivi”?

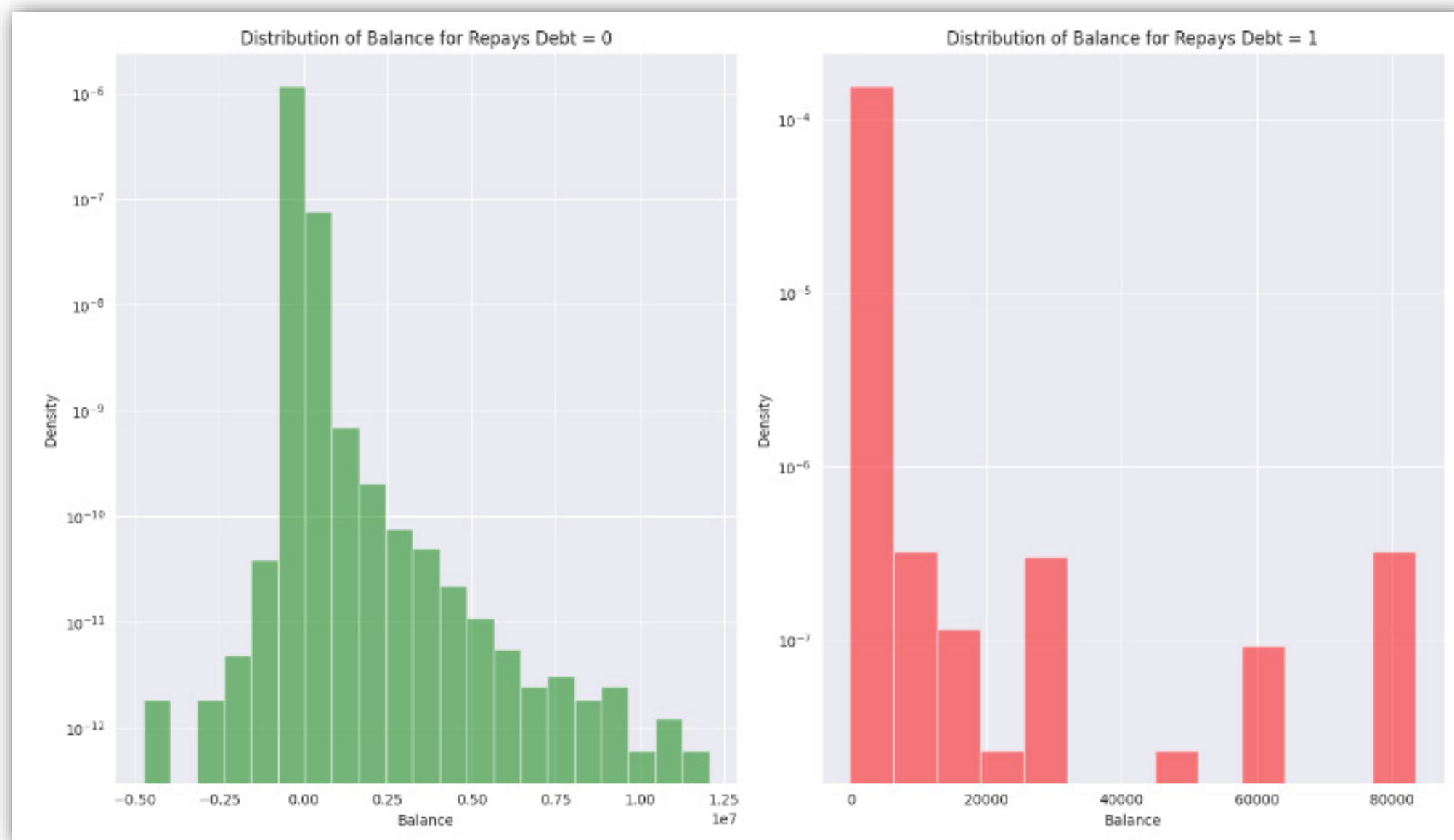


Notiamo che la colonna target risulta parecchio sbilanciata! Questo potrebbe portare ad un modello che non riesce a distinguere la classe minoritaria. In particolare, si nota che: la classe Yes (*repays_debt*=0), ossia i buoni clienti, è circa 306 volte più grande dell'altra classe.

Trovata la prima anomalia nel dataset abbiamo deciso di scavare più a fondo. Dobbiamo comprendere quali sono le features che incidono maggiormente nel determinare la bontà di un cliente.

SCAVIAMO PIÙ A FONDO

Abbiamo realizzato un istogramma di *balance* raggruppando in base a *repays_debt*.

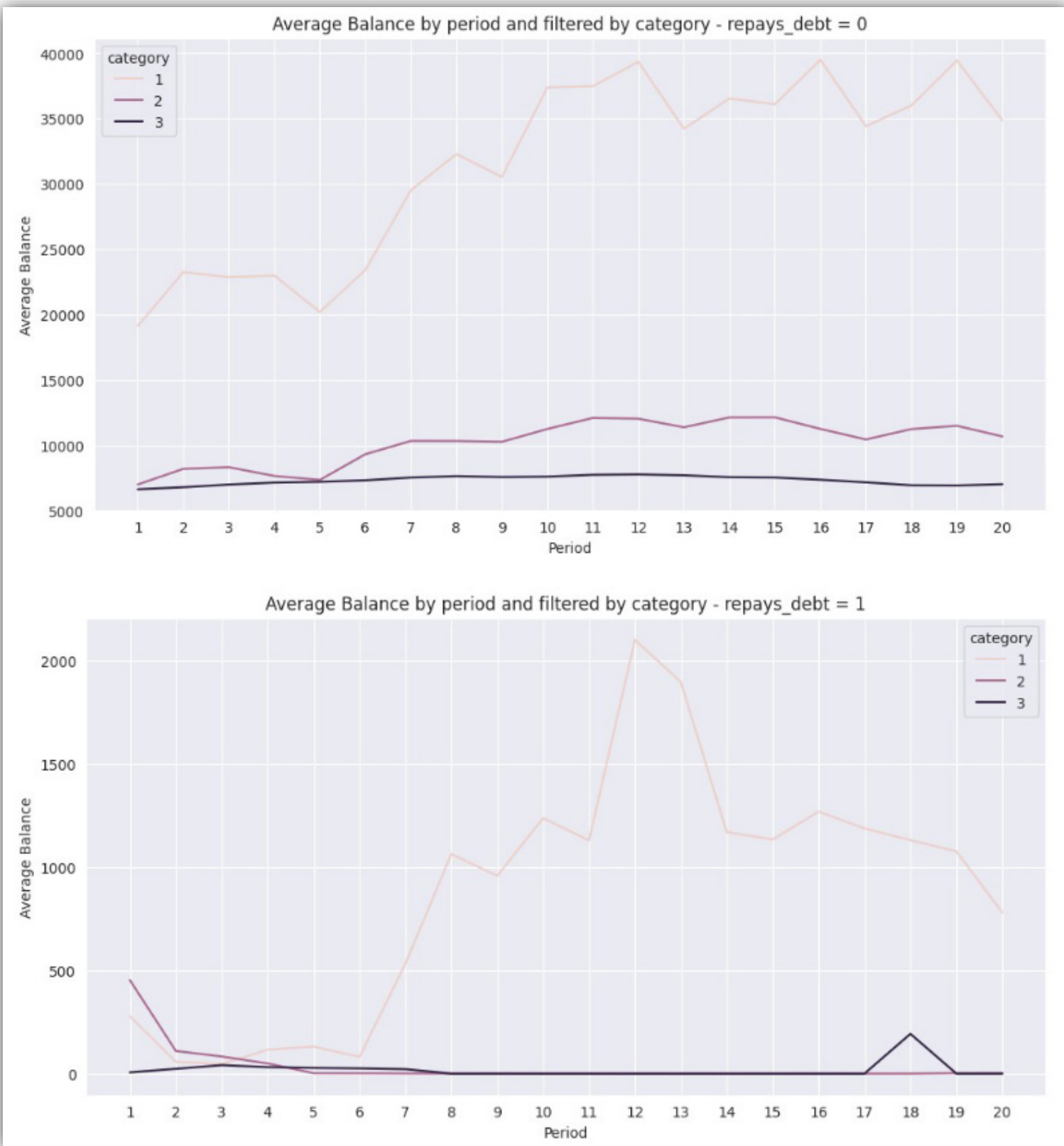


Per i buoni clienti i valori di *balance* risiedono in un range molto più ampio rispetto ai cattivi clienti. Ciò suggerisce una gestione attiva del conto e potenzialmente una migliore capacità di rimborso. Inoltre, i cattivi clienti hanno spesso saldo pressoché nullo. Questi due fatti indicano, per i cattivi clienti, una scarsa attività e un maggior rischio di insolvenza.

Si sottolinea nuovamente lo squilibrio tra le classi. Quindi la necessità di strategie mirate per gestire i diversi profili di rischio.

LAMPO DI GENIO?

Verifichiamo l'andamento medio nel tempo di *balance* per valutare la nostra ipotesi.



```
for val in df['repays_debt'].unique():
    df_filtered = df[df['repays_debt'] == val]

    # Average Balance for period and category
    average_balance = df_filtered.groupby(['period', 'category'])
    ['balance'].mean().reset_index()
    plt.figure(figsize=(12, 6))
    sns.lineplot(data=average_balance, x='period', y='balance',
    hue='category')

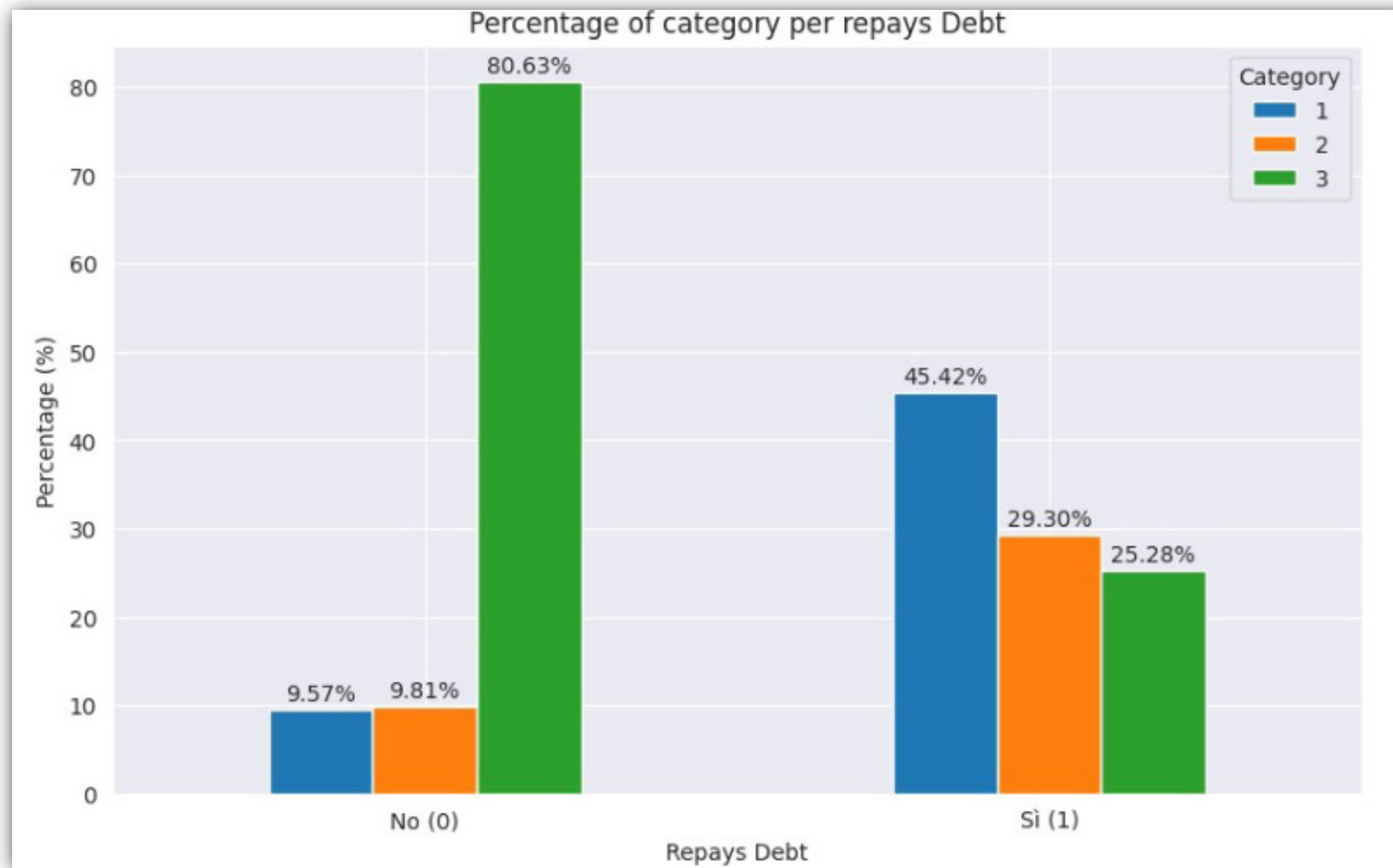
    plt.xlabel('Period')
    plt.ylabel('Average Balance')
    plt.title(f'Average Balance by period and filtered by category -
    repays_debt = {val}')
    plt.xticks(ticks=df['period'].unique())
    plt.show()
```

Confermando l'ipotesi iniziale, saldi costantemente a zero sono indicativi di incapacità di rimborso, indipendentemente dalla categoria. I conti aziendali (categoria 1), pur non mostrando saldi nulli, presentano valori atipici e un range differente rispetto ai buoni clienti.

L'appartenenza a una categoria specifica implica una maggiore probabilità di rimborsare il debito?

NON TUTTO ORO QUELLO CHE LUCCICA

A questo punto ci chiediamo se l'appartenenza ad una determinata categoria implichi una maggiore probabilità di rimborsare il debito o meno.



I buoni clienti appartengono per lo più alla categoria 3. Si potrebbe quasi dedurre una correlazione tra l'appartenenza alla categoria 3 e l'essere un buon cliente.


I cattivi clienti, invece, presentano una distribuzione senza picchi significativi.

Pertanto, non possiamo trarre alcuna conclusione.

Addestramento e risultati Task 1

La parte più importante del data preprocessing è il bilanciamento dei dati.

Come già detto questa operazione è necessaria in quanto il modello risultante non sarebbe in grado di distinguere la classe minoritaria.



```
undersampler = RandomUnderSampler(random_state=42)
X_train_resampled, y_train_resampled =
undersampler.fit_resample(X_train, y_train)
```

Tuttavia, in questo modo, il modello verrebbe addestrato solo su una piccola parte dei dati. Per cui abbiamo scelto di utilizzare diversi Random Forest addestrati su campioni diversi del dataset di partenza, impostando un *random_state* diverso per ogni modello. Infine, questi modelli vengono messi insieme utilizzando un VotingClassifier in modalità “soft”, in modo da ottenere prestazioni migliori assegnando più peso ai voti più importanti.

```
for i in range(n_models):
    undersampler = RandomUnderSampler(random_state=42 + i)
    X_train_resampled, y_train_resampled = undersampler.fit_resample(X_train,
    y_train)

    scaler = StandardScaler()
    X_train_resampled_standard = scaler.fit_transform(X_train_resampled)

    # Use the Random Forest with the best parameters we obtain until now
    model = RandomForestClassifier(random_state=1177 + i, max_depth=70,
    max_features="log2", min_samples_split=9, n_estimators=1000)
    model.fit(X_train_resampled_standard, y_train_resampled)

    models.append(('rf' + str(i), model))
```

Questo approccio ci ha consentito di raggiungere un F1 score pari a 0.7199.

Addestramento e risultati Task 2

Grazie al modello precedente possiamo individuare le features che hanno maggiore importanza. Quindi utilizziamo solo quelle assieme a *period*.

```
Important features:  
category_3: 0.20076629380709962  
client_id: 0.17928930313822078  
balance: 0.10424168572314513
```



```
train_data_2 = train_df_2[["period", "category", "client_id", "repays_debt"]]  
test_data_2 = test_df_2[["period", "category", "client_id", "repays_debt"]]
```

Anche qui si effettua il bilanciamento dei dati e, vista la ridotta dimensionalità del dataset, abbiamo preferito utilizzare un modello meno sofisticato: uno SGDClassifier che, per come è impostato, risulta equivalente ad una SVM avente kernel lineare.

```
modelS = SGDClassifier(random_state=177, n_jobs=-1)  
  
modelS.fit(X_train_resampled_standard_2, y_train_resampled_2)  
y_pred_2 = modelS.predict(X_test_standard_2)  
f1 = f1_score(y_test_2, y_pred_2)
```

Questo approccio ci ha consentito di raggiungere un F1 score pari a 0.6098.

*Ringraziamo lo Staff di Open Data Playground
e gli Sponsor per la grande opportunità*

Organizers

opendata
PLAYGROUND

DATA
SCIENCE | CINI
NATIONAL
LAB


Consiglio Nazionale
delle **Ricerche**

 UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARthenOPE

 UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

 **unipg**
A.D. 1308
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

 **UNIVERSITÀ**
DI PISA

Technical Partner

 **Palantir**

 **FOURTH AGE**

 **HPC4AI**

Data Provider



Sponsor



Educational Partner

 **Fastweb Digital**
Academy

HR Partner

OPEN SEARCH
NETWORK

Sustainability Partner

 **RIFORESTAZIONE**
AD ALTO
IMPATTO SOCIALE

With the patronage of

 **consorzio**
interuniversitario
nazionale
per l'informatica

REGIONE
TOSCANA
