```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
```

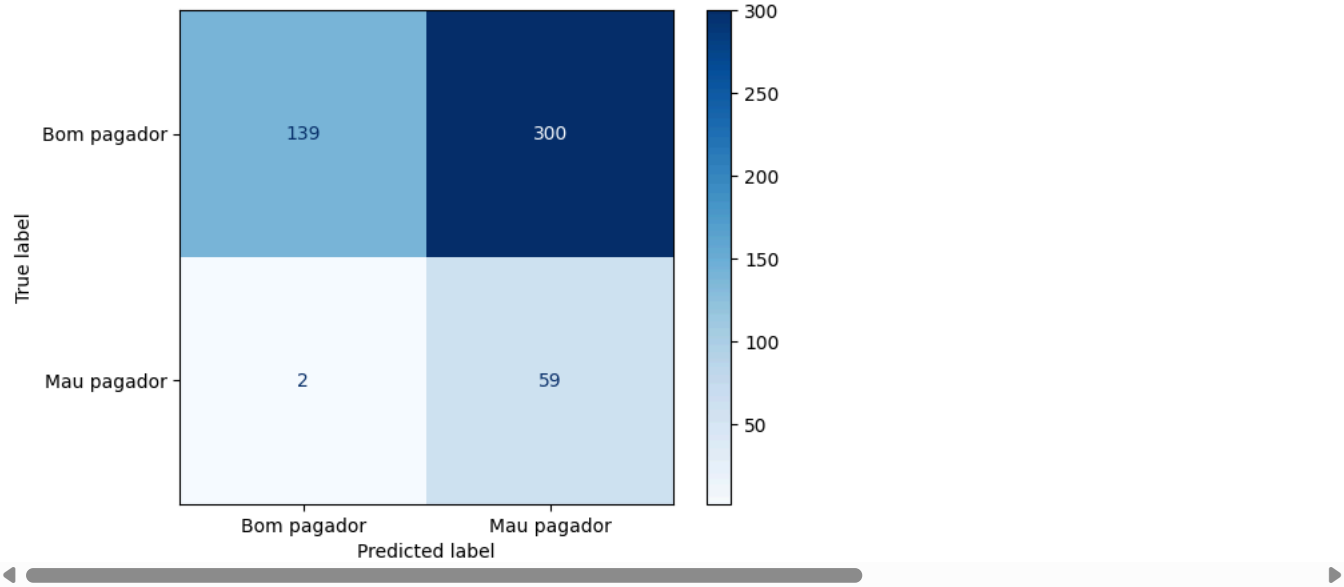
```
import pandas as pd
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


df = pd.read_csv("base_final.csv")

df.head()
```

	user_id	default_flag	classificacao_inadimplencia	more_90_days_overdue	number_times_delayed_payment_loan_30_59_days	number_ti
0	865	0	Bom pagador	98	98	
1	9884	0	Bom pagador	0	0	
2	18876	0	Bom pagador	0	0	
3	3108	0	Bom pagador	98	98	
4	14999	0	Bom pagador	0	0	

5 rows × 33 columns

```
y_real = df["default_flag"]
y_previsto = df["risco_dummy"]

matriz = confusion_matrix(y_real, y_previsto)

disp = ConfusionMatrixDisplay(confusion_matrix=matriz, display_labels=["Bom pagador", "Mau pagador"])
disp.plot(cmap='Blues')
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7d5fda1bdf90>
```




```
from sklearn.metrics import accuracy_score, precision_score, recall_score

print("Acurácia:", accuracy_score(y_real, y_previsto))
print("Precisão:", precision_score(y_real, y_previsto))
print("Recall (Sensibilidade):", recall_score(y_real, y_previsto))
```

```
Acurácia: 0.396
Precisão: 0.16434540389972144
```

```
Recall (Sensibilidade): 0.9672131147540983
```

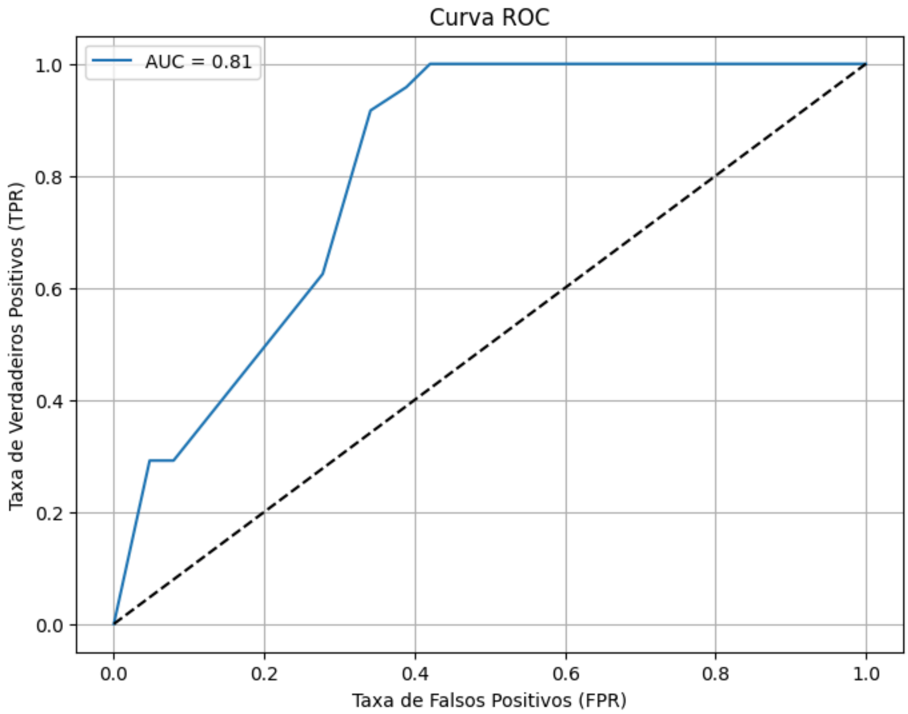
```
import pandas as pd
from sklearn.metrics import confusion_matrix, precision_score, recall_score, accuracy_score

df = pd.read_csv('base_final.csv')

def avaliar_cutoffs(df, max_score):
    resultados = []

    for corte in range(0, max_score + 1):
        df['predito'] = df['score'].apply(lambda x: 1 if x >= corte else 0)

        y_true = df['default_flag']
        y_pred = df['predito']

        vp = ((y_true == 1) & (y_pred == 1)).sum()
        vn = ((y_true == 0) & (y_pred == 0)).sum()
        fp = ((y_true == 0) & (y_pred == 1)).sum()
        fn = ((y_true == 1) & (y_pred == 0)).sum()

        acc = accuracy_score(y_true, y_pred)
        prec = precision_score(y_true, y_pred, zero_division=0)
        rec = recall_score(y_true, y_pred)

        resultados.append({
            'corte': corte,
            'VP': vp,
            'VN': vn,
            'FP': fp,
            'FN': fn,
            'Acurácia': round(acc, 3),
            'Precisão': round(prec, 3),
            'Recall': round(rec, 3)
        })

    return pd.DataFrame(resultados)


resultados_cutoff = avaliar_cutoffs(df, max_score=6)
print(resultados_cutoff)
```

```
    corte  VP   VN   FP  FN  Acurácia  Precisão  Recall
0      0  61    0  439   0     0.122     0.122   1.000
1      1  61    0  439   0     0.122     0.122   1.000
2      2  61    0  439   0     0.122     0.122   1.000
3      3  61   34  405   0     0.190     0.131   1.000
4      4  59  139  300   2     0.396     0.164   0.967
5      5  39  324  115  22     0.726     0.253   0.639
6      6   0  437    2  61     0.874     0.000   0.000
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
import matplotlib.pyplot as plt

features = [
    'age_dummy', 'salary_dummy', 'total_loans_dummy', 'more_90_days_dummy',
    'using_lines_dummy', 'debt_ratio_dummy', 'score'
]
target = 'default_flag'


X = df[features]
y = df[target]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)


print("Matriz de Confusão (ponto de corte padrão 0.5):")
print(confusion_matrix(y_test, y_pred))
print("\nRelatório de Classificação:")
print(classification_report(y_test, y_pred))
```

```
y_prob = model.predict_proba(X_test)[:, 1]

corte = 0.6
y_pred_corte = (y_prob >= corte).astype(int)

print(f"\nMatriz de Confusão (ponto de corte = {corte}):")
print(confusion_matrix(y_test, y_pred_corte))
print("\nRelatório de Classificação (ponto de corte ajustado):")
print(classification_report(y_test, y_pred_corte))


fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)

plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label=f'AUC = {auc:.2f}')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('Taxa de Falsos Positivos (FPR)')
plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')
plt.title('Curva ROC')
plt.legend()
plt.grid(True)
plt.show()
```

```
Matriz de Confusão (ponto de corte padrão 0.5):
[[126   0]
 [ 24   0]]

Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       126
           1       0.00      0.00      0.00        24

    accuracy                           0.84       150
   macro avg       0.42      0.50      0.46       150
weighted avg       0.71      0.84      0.77       150


Matriz de Confusão (ponto de corte = 0.6):
[[126   0]
 [ 24   0]]

Relatório de Classificação (ponto de corte ajustado):
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       126
           1       0.00      0.00      0.00        24

    accuracy                           0.84       150
   macro avg       0.42      0.50      0.46       150
weighted avg       0.71      0.84      0.77       150

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

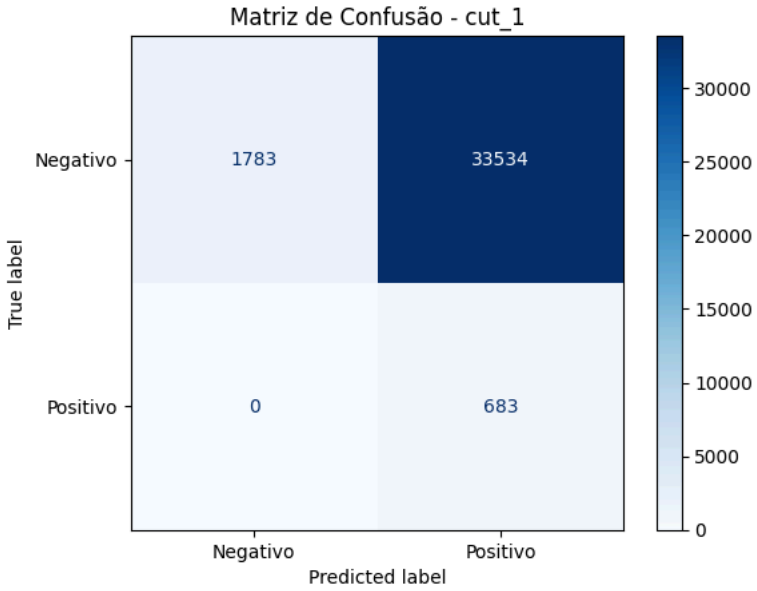


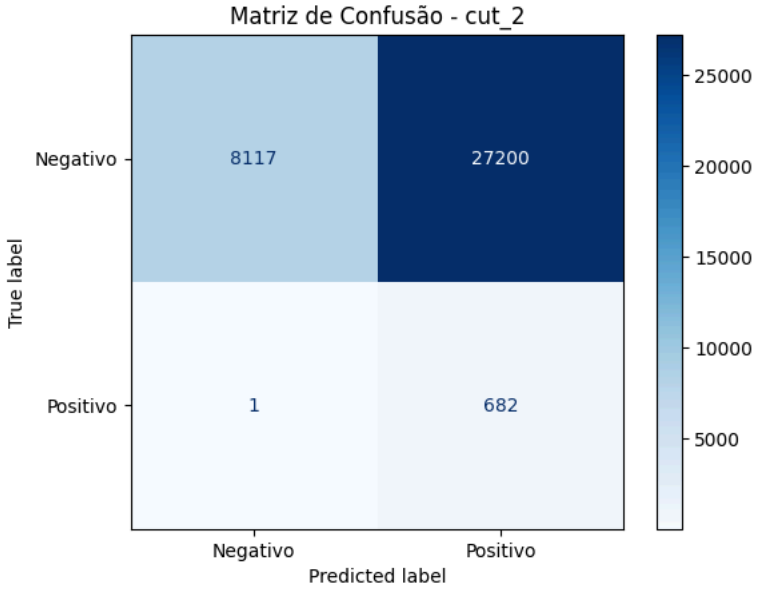

```
import pandas as pd
from sklearn.metrics import confusion_matrix, classification_report


df['predicted_default'] = (df['score'] >= 4).astype(int)

y_true = df['default_flag']

y_pred = df['predicted_default']

cm = confusion_matrix(y_true, y_pred)
print("Matriz de Confusão (score >= 4):")
print(cm)
```

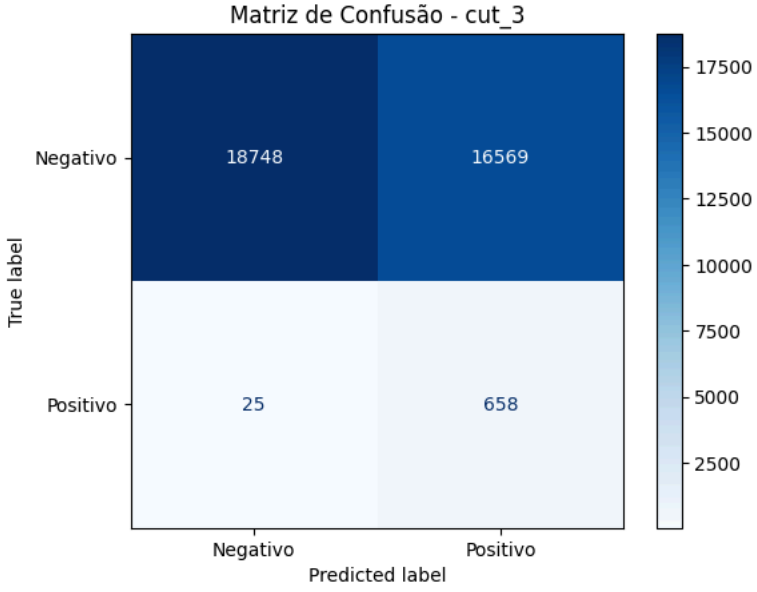
```
print("\nRelatório de Classificação:")
print(classification_report(y_true, y_pred))

accuracy = (cm[0,0] + cm[1,1]) / cm.sum()
print(f"Acurácia calculada: {accuracy:.3f}")
```

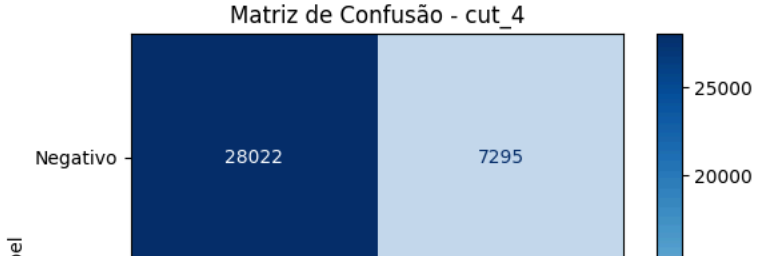
```
Matriz de Confusão (score >= 4):
[[139 300]
 [  2  59]]

Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.99      0.32      0.48       439
           1       0.16      0.97      0.28        61

    accuracy                           0.40       500
   macro avg       0.58      0.64      0.38       500
weighted avg       0.89      0.40      0.46       500

Acurácia calculada: 0.396
```

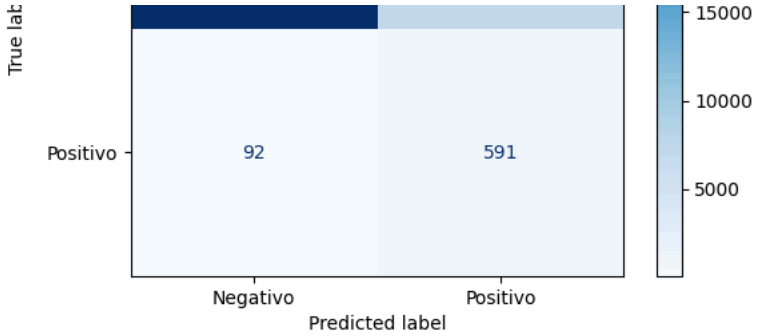
```
import pandas as pd
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

dados = {
    'corte':   ['cut_1', 'cut_2', 'cut_3', 'cut_4', 'cut_5', 'cut_6'],
    'VP':      [683, 682, 658, 591, 404, 78],   # Verdadeiro Positivo
    'VN':      [1783, 8117, 18748, 28022, 33227, 35101],  # Verdadeiro Negativo
    'FP':      [33534, 27200, 16569, 7295, 2090, 216],  # Falso Positivo
    'FN':      [0, 1, 25, 92, 279, 605]  # Falso Negativo
}

df = pd.DataFrame(dados)

for index, row in df.iterrows():
    corte = row['corte']
    VP = row['VP']
    VN = row['VN']
    FP = row['FP']
    FN = row['FN']

    y_true = [1]*VP + [1]*FN + [0]*FP + [0]*VN
    y_pred = [1]*VP + [0]*FN + [1]*FP + [0]*VN

    cm = confusion_matrix(y_true, y_pred)

    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Negativo", "Positivo"])
    disp.plot(cmap='Blues')
    plt.title(f'Matriz de Confusão - {corte}')
    plt.grid(False)
    plt.show()
```

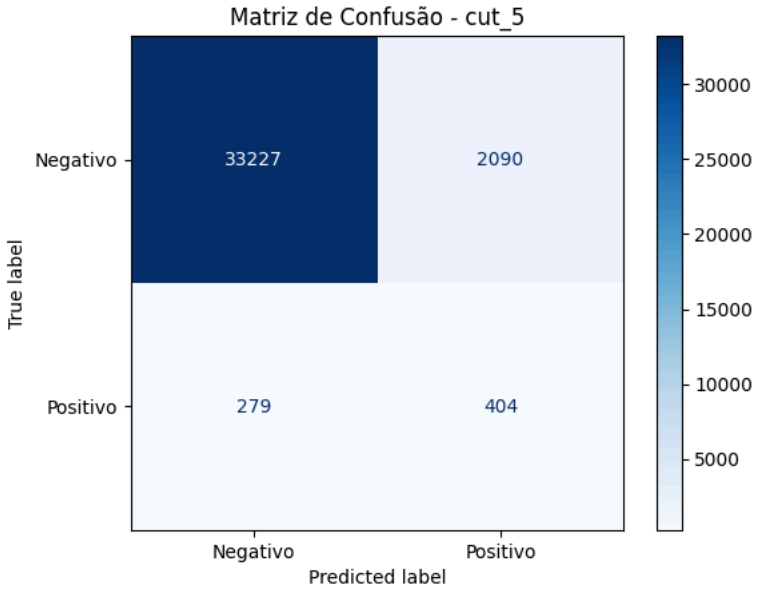

Matriz de Confusão - cut_1
Negativo
Positivo
True label
1783
33534
0
683
Negativo
Positivo
Predicted label
30000
25000
20000
15000
10000
5000
0

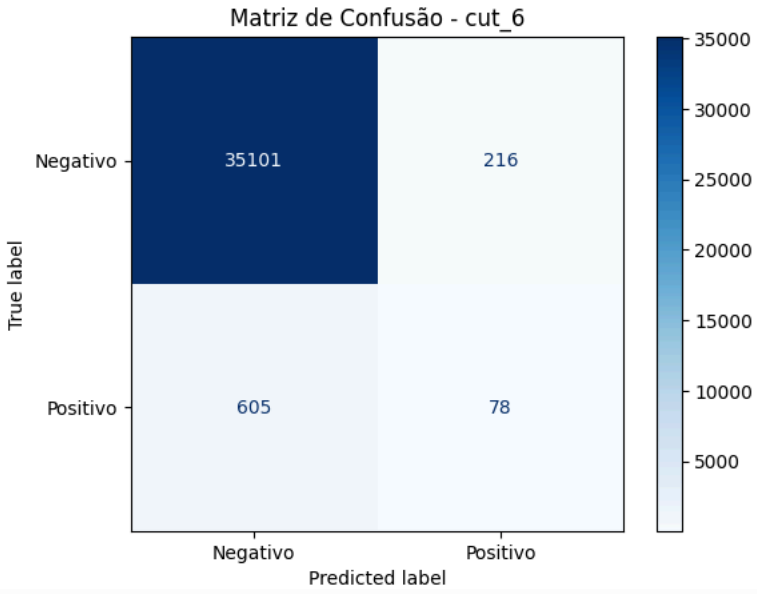

Matriz de Confusão - cut_2
Negativo
Positivo
True label
8117
27200
1
682
Negativo
Positivo
Predicted label
25000
20000
15000
10000
5000


Matriz de Confusão - cut_3
Negativo
Positivo
True label
18748
16569
25
658
Negativo
Positivo
Predicted label
17500
15000
12500
10000
7500
5000
2500


Matriz de Confusão - cut_4
Negativo
28022
7295
25000
20000


True lat
Positivo
92
591
Negativo
Positivo
Predicted label
15000
10000
5000


Matriz de Confusão - cut_5
Negativo
33227
2090
True label
Positivo
279
404
Negativo
Positivo
Predicted label
30000
25000
20000
15000
10000
5000


Matriz de Confusão - cut_6
Negativo
35101
216
True label
Positivo
605
78
Negativo
Positivo
Predicted label
35000
30000
25000
20000
15000
10000
5000

```
import pandas as pd

dados = {
    'corte': ['cut_1', 'cut_2', 'cut_3', 'cut_4', 'cut_5', 'cut_6'],
    'VP': [683, 682, 658, 591, 404, 781]
```