Kaggle Scraping

Describe background Gradient

`train_data.describe().T.style.background_gradient()`

Cmap

```
import matplotlib.colors as mcolors
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

PALETTE = ['#86FFBC', '#FF80FF', '#99FFFF', '#FF9999', '#CCFF66
sns.set(style="whitegrid")
sns.set_palette(PALETTE)
```

HeatMap

```
heatmap = sns.heatmap(conf_matrix, annot=True, fmt="d", cmap='vi
```

Missing Values Plot

```
sns.displot(data=train_data.isnull().melt(value_name='missin
g'),
    y='variable',
    hue='missing',
    multiple='fill',
    height=8,
```

```
#     width=10,aspect=1.6
)

# specifying a threshold valueplt.axvline(0.4, color='r')
plt.title('Null Values in Train Data', fontsize=13)
plt.show()
```

Y value count

```
f,ax=plt.subplots(1,2,figsize=(19,8))
train_data['Exited'].value_counts().plot.pie(autopct='%1.1f%
%',ax=ax[0],shadow=True)
# ax[0].set_title('Pie-Plot')ax[0].set_ylabel('')
sns.countplot(x='Exited',data=train_data,ax=ax[1])
# ax[1].set_title('Count-Plot')plt.suptitle('Target Value Ana
ysis - Competition Data')
plt.show()
```

Select Object

```
object_columns = df.select_dtypes(include=['object']).columns
```

Histplot for numericals

```
for column in continuous_vars:
    fig,ax =plt.subplots(figsize=(18, 4))
    fig =sns.histplot(data=train_data, x=column, hue="Exite
d", bins=50, kde=True)
    plt.show()


#Using the Palette:
```

```
fig = plt.subplots(1, 1, figsize=(12, 12), dpi=100)
sns.histplot(y, bins = [i for i in range(10)], color=PALETTE
[2], label='y_true')
sns.histplot(pred_test_all_rounds, bins = [i for i in range(1
0)], color=PALETTE[1], label='y_pred')
```

Categorical value plot

```
forcolumnincategorical_vars:
f,ax=plt.subplots(1,2,figsize=(18,5.5))
train_data[column].value_counts().plot.pie(autopct='%1.1f%%',
ax=ax[0],shadow=True)
ax[0].set_ylabel(f'{column}')
sns.countplot(x=column,data=train_data,ax=ax[1])
plt.suptitle(f'{column}')
plt.show()
```

Pairplot (Multivariate Analysis)

```
df3 =train_data[['CreditScore', 'Age', 'Balance', 'EstimatedS
alary', 'Exited']].copy()
sns.pairplot(df3, hue="Exited", corner=True)
plt.show()
```

XGboost

```
clf_xgb_v1 = xgb.XGBClassifier(objective='binary:logistic',
                               # missing=None,
                               seed=42)
```

```
clf_xgb_v1.fit(X_train,
            y_train,
            verbose=True,
            early_stopping_rounds=10,
            eval_metric='auc',
            eval_set=[(X_test, y_test)])
```

Confusion Matrix Display

```
cm = confusion_matrix(y_test, predictions_1)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labe
disp.plot()
plt.show()
```

GridsearchCV

```
# param_grid = {
#     'max_depth': [3, 4, 5],
#     'learning_rate': [0.05, 0.01, 0.1],
#     'gamma': [0, 0.25, 1.0],
#     'reg_lambda': [0, 1.0, 10.0],
#     'scale_pos_weight': [1, 3, 5]
# }

# optimal_parameters = GridSearchCV(
#     estimator=xgb.XGBClassifier(objective='binary:logistic',
#                                 seed=42,
#                                 subsample=0.9,
#                                 colsample_bytree=0.5),
#     param_grid=param_grid,
#     scoring='roc_auc',
#     verbose=3,
```

```
#     n_jobs=10,
#     cv=3
# )
```

Realtime visualization using matplotlib

```
import random
from itertools import count
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

plt.style.use('fivethirtyeight')

x_vals = []
y_vals = []

index = count()


def animate(i):
    data = pd.read_csv('data.csv')
    x = data['x_value']
    y1 = data['total_1']
    y2 = data['total_2']

    plt.cla()  #to clear the axis

    plt.plot(x, y1, label='Channel 1')
    plt.plot(x, y2, label='Channel 2')

    plt.legend(loc='upper left')
    plt.tight_layout()
```

```
ani = FuncAnimation(plt.gcf(), animate, interval=1000)

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