```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

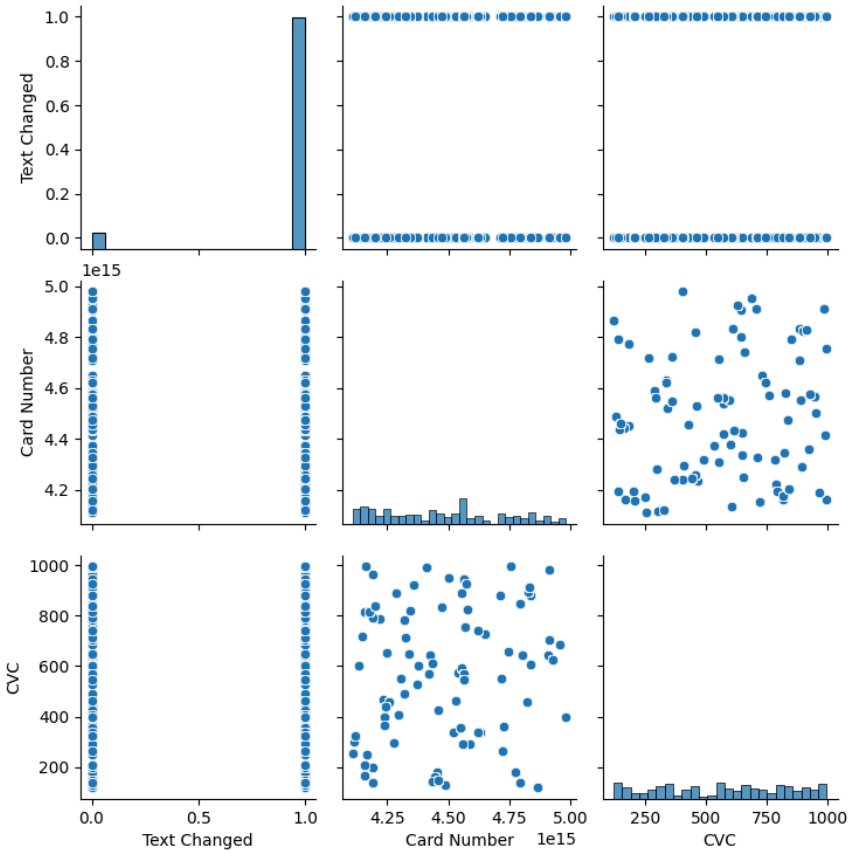
```
!ls "/content/drive/MyDrive/raw_kmt_dataset"
```

```
raw_kmt_user_0001.json  raw_kmt_user_0023.json  raw_kmt_user_0045.json  raw_kmt_user_0067.json
raw_kmt_user_0002.json  raw_kmt_user_0024.json  raw_kmt_user_0046.json  raw_kmt_user_0068.json
raw_kmt_user_0003.json  raw_kmt_user_0025.json  raw_kmt_user_0047.json  raw_kmt_user_0069.json
raw_kmt_user_0004.json  raw_kmt_user_0026.json  raw_kmt_user_0048.json  raw_kmt_user_0070.json
raw_kmt_user_0005.json  raw_kmt_user_0027.json  raw_kmt_user_0049.json  raw_kmt_user_0071.json
raw_kmt_user_0006.json  raw_kmt_user_0028.json  raw_kmt_user_0050.json  raw_kmt_user_0072.json
raw_kmt_user_0007.json  raw_kmt_user_0029.json  raw_kmt_user_0051.json  raw_kmt_user_0073.json
raw_kmt_user_0008.json  raw_kmt_user_0030.json  raw_kmt_user_0052.json  raw_kmt_user_0074.json
raw_kmt_user_0009.json  raw_kmt_user_0031.json  raw_kmt_user_0053.json  raw_kmt_user_0075.json
raw_kmt_user_0010.json  raw_kmt_user_0032.json  raw_kmt_user_0054.json  raw_kmt_user_0076.json
raw_kmt_user_0011.json  raw_kmt_user_0033.json  raw_kmt_user_0055.json  raw_kmt_user_0077.json
raw_kmt_user_0012.json  raw_kmt_user_0034.json  raw_kmt_user_0056.json  raw_kmt_user_0078.json
raw_kmt_user_0013.json  raw_kmt_user_0035.json  raw_kmt_user_0057.json  raw_kmt_user_0079.json
raw_kmt_user_0014.json  raw_kmt_user_0036.json  raw_kmt_user_0058.json  raw_kmt_user_0080.json
raw_kmt_user_0015.json  raw_kmt_user_0037.json  raw_kmt_user_0059.json  raw_kmt_user_0081.json
raw_kmt_user_0016.json  raw_kmt_user_0038.json  raw_kmt_user_0060.json  raw_kmt_user_0082.json
raw_kmt_user_0017.json  raw_kmt_user_0039.json  raw_kmt_user_0061.json  raw_kmt_user_0083.json
raw_kmt_user_0018.json  raw_kmt_user_0040.json  raw_kmt_user_0062.json  raw_kmt_user_0084.json
raw_kmt_user_0019.json  raw_kmt_user_0041.json  raw_kmt_user_0063.json  raw_kmt_user_0085.json
raw_kmt_user_0020.json  raw_kmt_user_0042.json  raw_kmt_user_0064.json  raw_kmt_user_0086.json
raw_kmt_user_0021.json  raw_kmt_user_0043.json  raw_kmt_user_0065.json  raw_kmt_user_0087.json
raw_kmt_user_0022.json  raw_kmt_user_0044.json  raw_kmt_user_0066.json  raw_kmt_user_0088.json
```

```
import pandas as pd
import json
import os
finaldf=[]
directory = "/content/drive/MyDrive/raw_kmt_dataset"
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    with open(f, "r") as read_file:
      obj = json.load(read_file)
      pretty_json = json.dumps(obj, indent=4)
    details_df = pd.DataFrame.from_dict([obj["details"]])
    key_events = obj["true_data"]["test_1"]["key_events"]
    key_events2=obj["false_data"]["test_1"]["key_events"]
    events_df = pd.DataFrame(key_events)
    events_df["Clickstream Type"]="True Data"
    events_df1=pd.DataFrame(key_events2)
    events_df1["Clickstream Type"]="False_Data"
    df=pd.concat([events_df1,events_df])
    df["ID"]=details_df["ID"]
    df["Provider"]=details_df["Provider"]
    df["Name"]=details_df["Name"]
    df["Card Number"]=details_df["Card Number"]
    df["CVC"]=details_df["CVC"]
    df["Expiry"]=details_df["Expiry"]
    x1=details_df["ID"][0]
    x2=details_df["Provider"][0]
    x3=details_df["Name"][0]
    x4=details_df["Card Number"][0]
    x5=details_df["CVC"][0]
    x6=details_df["Expiry"][0]
    df["ID"]=df["ID"].fillna(x1)
    df["Provider"]=df["Provider"].fillna(x2)
    df["Name"]=df["Name"].fillna(x3)
    df["Card Number"]=df["Card Number"].fillna(x4)
    df["CVC"]=df["CVC"].fillna(x5)
    df["Expiry"]=df["Expiry"].fillna(x6)
    finaldf.append(df)
```

```
import pandas as pd
result_df = pd.concat(finaldf, ignore_index=True)
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(result_df)
plt.show()
```




```
result_df.shape
df.head(10)
```

	Key	Event	Input Box	Text Changed	Timestamp	Epoch	Clickstream Type
0	shift	pressed	Name	True	2022-03-24 19:05:11.614202	1648148711.614202	False_Data
1	m	pressed	Name	True	2022-03-24 19:05:11.716469	1648148711.716469	False_Data
2	m	released	Name	True	2022-03-24 19:05:11.767782	1648148711.7677824	False_Data
3	shift	released	Name	True	2022-03-24 19:05:11.791855	1648148711.7918556	False_Data
4	r	pressed	Name	True	2022-03-24 19:05:11.791855	1648148711.7918556	False_Data
5	r	released	Name	True	2022-03-24 19:05:11.867780	1648148711.8677807	False_Data
6	spacebar	pressed	Name	True	2022-03-24 19:05:11.905833	1648148711.9058332	False_Data

Next steps: Generate code with df View recommended plots

```
result_df['Input Box'].unique()
result_df['Input Box'].value_counts()
```

```
Input Box
Name       8534
```

```
Card No    6585
CVC        1151
Exp m       880
Exp y       810
Null         58
Name: count, dtype: int64
```

```
result_df['Event'].value_counts()
```

```
Event
pressed     9186
released    8832
Name: count, dtype: int64
```

```
result_df.shape
```

```
(18018, 13)
```

```
result_df["Name"].unique()
```

```
array(['Ms Lily Watson', 'Miss Sofia Morris', 'Ms Lucy Jackson',
       'Mrs Myla Ellis', 'Ms Heidi Owen', 'Mr William Cook',
       'Mr Austin Hughes', 'Mr Jude Campbell', 'Mr Jude Taylor',
       'Miss Luna Watson', 'Mr Samuel Fisher', 'Mrs Freya Jackson',
       'Ms Thea Cook', 'Mrs Ellie Miller', 'Mr James Knight',
       'Mr Benjamin Collins', 'Ms Ivy Collins', 'Mr Luke Martin',
       'Ms Holly Murray', 'Mrs Freya Foster', 'Mr Kai Kelly',
       'Miss Mia Davies', 'Mrs Darcie Miller', 'Mrs Robyn Foster',
       'Ms Heidi Jones', 'Miss Lily Roberts', 'Mr Mason Rogers',
       'Mr Jaxon James', 'Mrs Luna Wilson', 'Mr Harley Wilkinson',
       'Mr Bobby Clark', 'Mr James Ellis', 'Mrs Matilda Walker',
       'Mr William Bell', 'Mr Louis Young', 'Ms Harper Wood',
       'Mr Ezra Simpson', 'Miss Chloe Turner', 'Mrs Esmae Brown',
       'Mrs Clara Thomson', 'Mr Jenson Murphy', 'Ms Georgia Carter',
       'Mrs Arabella Pearson', 'Miss Anna Watson', 'Miss Alice Davies',
       'Mrs Iris Knight', 'Miss Jessica Bell', 'Ms Ivy Lee',
       'Mr Edward Marshall', 'Mr Oakley Richards', 'Mr Louis Baker',
       'Mrs Eliza Robertson', 'Miss Evie Hunt', 'Miss Hannah Stewart',
       'Mrs Felicity Martin', 'Mr Luca Brown', 'Miss Lilly Williams',
       'Mrs Phoebe Reid', 'Mr Bobby Cook', 'Mr Reggie Mitchell',
       'Mrs Elsie Adams', 'Mr Louie Chapman', 'Mr Felix Hughes',
       'Ms Eliza Marshall', 'Mr Ethan Bailey', 'Mrs Maisie James',
       'Miss Ava Clark', 'Mr Roman Robertson', 'Mr Samuel Mason',
       'Mr Hudson Johnson', 'Mr Jaxon Butler', 'Mr Rory Griffiths',
       'Miss Esmae Hunt', 'Miss Chloe Bailey', 'Mr Pene Richards',
       'Mr Isaac Richards', 'Mrs Evie Thomson', 'Mr Frankie Knight',
       'Mr Myles Stevens', 'Mrs Esmae Morris', 'Mr Vinnie Gray',
       'Mr Elliot Lewis', 'Mr Sonny Hall', 'Mr Theo Kelly',
       'Ms Hannah Rogers', 'Mrs Sofia Wright', 'Mr Max Rogers'],
      dtype=object)
```

```
del result_df["Timestamp"]
```

```
result_df.head(10)
```

	Key	Event	Input Box	Text Changed	Epoch	Clickstream Type	ID	Prc
0	shift	pressed	Name	True	1645967849.524535	False_Data	CDID0014	Mast
1	m	pressed	Name	True	1645967849.8765113	False_Data	CDID0014	Mast
2	shift	released	Name	True	1645967849.975332	False_Data	CDID0014	Mast
3	m	released	Name	True	1645967850.0119765	False_Data	CDID0014	Mast
4	s	pressed	Name	True	1645967850.1334922	False_Data	CDID0014	Mast
5	s	released	Name	True	1645967850.2678561	False_Data	CDID0014	Mast

Next steps: Generate code with result_df View recommended plots

```
result_df["Target"]=result_df["Clickstream Type"]
```

```
del result_df["Clickstream Type"]
```

```
result_df
```

	Key	Event	Input Box	Text Changed	Epoch	ID	Provider	
0	shift	pressed	Name	True	1645967849.524535	CDID0014	MasterCard	M W
1	m	pressed	Name	True	1645967849.8765113	CDID0014	MasterCard	M W
2	shift	released	Name	True	1645967849.975332	CDID0014	MasterCard	M W
3	m	released	Name	True	1645967850.0119765	CDID0014	MasterCard	M W
4	s	pressed	Name	True	1645967850.1334922	CDID0014	MasterCard	M W
...	...	...	...	...	...	...	...	
18013	numpad8	released	Exp m	False	1648153098.743069	CDID0079	Discover	M R
								M

Next steps: Generate code with `result_df` View recommended plots

```
result_df.isnull().sum()
#as there are no null values, data cleaning is not needed
```

```
Key             0
Event           0
Input Box       0
Text Changed    0
Epoch           0
ID              0
Provider        0
Name            0
Card Number     0
CVC             0
Expiry          0
Target          0
dtype: int64
```

```
result_df.columns
```

```
Index(['Key', 'Event', 'Input Box', 'Text Changed', 'Epoch', 'ID', 'Provider',
       'Name', 'Card Number', 'CVC', 'Expiry', 'Target'],
      dtype='object')
```

```
target = result_df['Target']
df_features = result_df.drop(columns=['Target'])
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for x in result_df.columns:
  result_df[x]=le.fit_transform(result_df[x])
df3=result_df
#while training the dataset using XGBoost and SVM, object datatype is not accepted
```

```
df3.dtypes
```

```
Key             int64
Event           int64
Input Box       int64
Text Changed    int64
Epoch           int64
ID              int64
Provider        int64
Name            int64
Card Number     int64
CVC             int64
Expiry          int64
Target          int64
dtype: object
```

```
#Given the features of the dataset, outlier detection is not necessary.
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
df_subset = df3.sample(frac=0.1, random_state=42)

#due to large data size, I have tried random sampling to see how it works. Due to need of more storage I shifted to kaggle
# to use its storage and GPU power. The accuracy remains the same even when applying it to the entire dataset.

independent=['Key', 'Event', 'Input Box', 'Text Changed', 'Epoch', 'ID', 'Provider', 'Name', 'Card Number', 'CVC', 'Expiry']
X_subset = df_subset[independent]
y_subset = df_subset['Target']
X_train_subset, X_test_subset, y_train_subset, y_test_subset = train_test_split(
    X_subset, y_subset, test_size=0.2, random_state=42)


rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_subset, y_train_subset)
rf_predictions = rf_model.predict(X_test_subset)

print("Random Forest:")
print("Accuracy:", accuracy_score(y_test_subset, rf_predictions))
print("Classification Report:\n", classification_report(y_test_subset, rf_predictions))
```

```
Random Forest:
Accuracy: 0.9916897506925207
Classification Report:
                precision    recall  f1-score   support

           0       0.99      0.99      0.99       168
           1       0.99      0.99      0.99       193

    accuracy                           0.99       361
   macro avg       0.99      0.99      0.99       361
weighted avg       0.99      0.99      0.99       361
```

```
import xgboost as xgb

xgb_model = xgb.XGBClassifier(n_estimators=100, random_state=42)
xgb_model.fit(X_train_subset, y_train_subset)
xgb_predictions = xgb_model.predict(X_test_subset)

print("\nXGBoost:")
print("Accuracy:", accuracy_score(y_test_subset, xgb_predictions))
print("Classification Report:\n", classification_report(y_test_subset, xgb_predictions))
```

```
XGBoost:
Accuracy: 1.0
Classification Report:
                precision    recall  f1-score   support

           0       1.00      1.00      1.00       168
           1       1.00      1.00      1.00       193

    accuracy                           1.00       361
   macro avg       1.00      1.00      1.00       361
weighted avg       1.00      1.00      1.00       361
```

```
svm_model = SVC(random_state=42)
svm_model.fit(X_train_subset, y_train_subset)
svm_predictions = svm_model.predict(X_test_subset)

print("\nSVM:")
print("Accuracy:", accuracy_score(y_test_subset, svm_predictions))
print("Classification Report:\n", classification_report(y_test_subset, svm_predictions))
```

```
SVM:
Accuracy: 0.6232686980609419
Classification Report:
                precision    recall  f1-score   support

           0       0.56      0.92      0.69       168
           1       0.84      0.37      0.51       193

    accuracy                           0.62       361
   macro avg       0.70      0.64      0.60       361
weighted avg       0.71      0.62      0.60       361
```