Mục đích xây dựng mô hình: Dự đoán đơn hàng có xảy ra sự cố hay không và tìm ra càng nhiều đơn hàng xảy ra sự cố -> tập trung vào Recall class positive (+) -> tăng Recall cao nhất có thể

Class positive (+): đơn hàng xảy ra sự cố

Class negative (-): đơn hàng không xảy ra sự cố

Danh sách thành viên trong nhóm:

Mai Ngọc Hoàng 22139023

Trương Lý Minh Hoàng 22139025

Trần Minh Hữu 22139028

Lê Huỳnh Đức 22139017

Võ Văn Hiếu 22139021

Steps for building model machine learning

1. Data collection

In [1]:
```
import pandas as pd
import numpy as np
df = pd.read_csv("clean_feature(1).csv")
df["Outcome"] = np.where((df["period"] > 63) | (df["passed"] < df["count"]), 1, 0)
df
```

Out[1]:

	Day	DoWeek	hour	count	passed	period	data	ServiceID	C
0	12/29/2021	2	17	2974	2974	63.952589	612.081036	2	
1	12/29/2021	2	17	4528	4528	63.000000	960.307862	11	
2	12/29/2021	2	17	19	19	62.000000	566.315790	5	
3	12/29/2021	2	17	1207	1207	63.000000	601.025684	7	
4	12/29/2021	2	18	8450	8450	63.942130	611.024024	2	
...	...	...	...	...	...	...	...	...	
450666	7/24/2023	0	18	1	1	62.000000	624.000000	1	
450667	7/24/2023	0	18	108	108	62.000000	1075.972222	5	
450668	7/24/2023	0	18	5279	5279	63.000000	599.900170	7	
450669	7/24/2023	0	18	2	2	63.000000	1437.500000	0	
450670	7/24/2023	0	18	18	18	0.000000	378255.166700	10	

450671 rows × 9 columns

In [2]:
```
df.columns
```

Out[2]:
```
Index(['Day', 'DoWeek', 'hour', 'count', 'passed', 'period', 'data',
       'ServiceID', 'Outcome'],
      dtype='object')
```

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

Out[3]:

	DoWeek	hour	count	passed	period	
count	450671.000000	450671.000000	450671.000000	450671.000000	450671.000000	450671.
mean	2.813627	12.357582	3177.112321	3176.849074	61.856996	6273.
std	1.944569	5.960330	4541.786519	4541.751962	7.566910	41830.
min	0.000000	0.000000	1.000000	0.000000	0.000000	0.
25%	1.000000	8.000000	33.000000	33.000000	62.000000	599.
50%	3.000000	13.000000	768.000000	767.000000	63.000000	645.
75%	4.000000	17.000000	4879.000000	4878.000000	63.000000	955.
max	6.000000	23.000000	26273.000000	26273.000000	64.000300	378361.

2. Statistics

In [4]:
```
df.head(10)
```

Out[4]:

	Day	DoWeek	hour	count	passed	period	data	ServiceID	Outcome
0	12/29/2021	2	17	2974	2974	63.952589	612.081036	2	1
1	12/29/2021	2	17	4528	4528	63.000000	960.307862	11	0
2	12/29/2021	2	17	19	19	62.000000	566.315790	5	0
3	12/29/2021	2	17	1207	1207	63.000000	601.025684	7	0
4	12/29/2021	2	18	8450	8450	63.942130	611.024024	2	1
5	12/29/2021	2	18	13101	13101	63.000000	947.268377	11	0
6	12/29/2021	2	18	115	115	62.000000	2922.947826	5	0
7	12/29/2021	2	18	3546	3546	63.000000	600.478003	7	0
8	12/29/2021	2	19	5743	5743	63.933658	605.911196	2	1
9	12/29/2021	2	19	8863	8863	63.000000	940.703712	11	0

In [5]:
```
df.dtypes
```

Out[5]:
```
Day           object
DoWeek         int64
hour           int64
count          int64
passed         int64
period       float64
data         float64
ServiceID      int64
Outcome        int64
dtype: object
```

In [6]:
```
df.describe()
```

Out[6]:

	DoWeek	hour	count	passed	period	
count	450671.000000	450671.000000	450671.000000	450671.000000	450671.000000	450671.
mean	2.813627	12.357582	3177.112321	3176.849074	61.856996	6273.
std	1.944569	5.960330	4541.786519	4541.751962	7.566910	41830.
min	0.000000	0.000000	1.000000	0.000000	0.000000	0.
25%	1.000000	8.000000	33.000000	33.000000	62.000000	599.
50%	3.000000	13.000000	768.000000	767.000000	63.000000	645.
75%	4.000000	17.000000	4879.000000	4878.000000	63.000000	955.
max	6.000000	23.000000	26273.000000	26273.000000	64.000300	378361.

3. Data preprocessing

3.1 Handle missing or invalid values

In [7]:
```
#Check missing datas
df.isna().sum()
```

Out[7]:
```
Day          0
DoWeek       0
hour         0
count        0
passed       0
period       0
data         0
ServiceID    0
Outcome      0
dtype: int64
```

In [8]:
```
#Check duplicated datas
df.loc[df.duplicated()]
```

Out[8]:

	Day	DoWeek	hour	count	passed	period	data	ServiceID	Outcom
2021	1/11/2022	1	18	1	1	63.0	877.000000	0	
2764	1/16/2022	6	13	1	1	62.0	301.000000	1	
3731	1/22/2022	5	10	1	1	63.0	947.000000	0	
4620	1/27/2022	3	16	1	1	63.0	945.000000	0	
4919	1/29/2022	5	10	2	2	62.0	301.000000	1	
...	...	...	...	...	...	...	...	...	
450620	7/17/2023	0	18	1	1	62.0	624.000000	1	
450637	7/19/2023	2	18	3	3	62.0	339.666667	8	
450645	7/21/2023	4	17	8	8	62.0	514.500000	8	
450652	7/21/2023	4	18	4	4	62.0	334.250000	8	
450660	7/24/2023	0	17	1	1	62.0	383.000000	8	

7660 rows × 9 columns

In [9]:
```
#Drop duplicated datas
df= df.loc[~df.duplicated()].reset_index(drop=True).copy()
df
```

Out[9]:

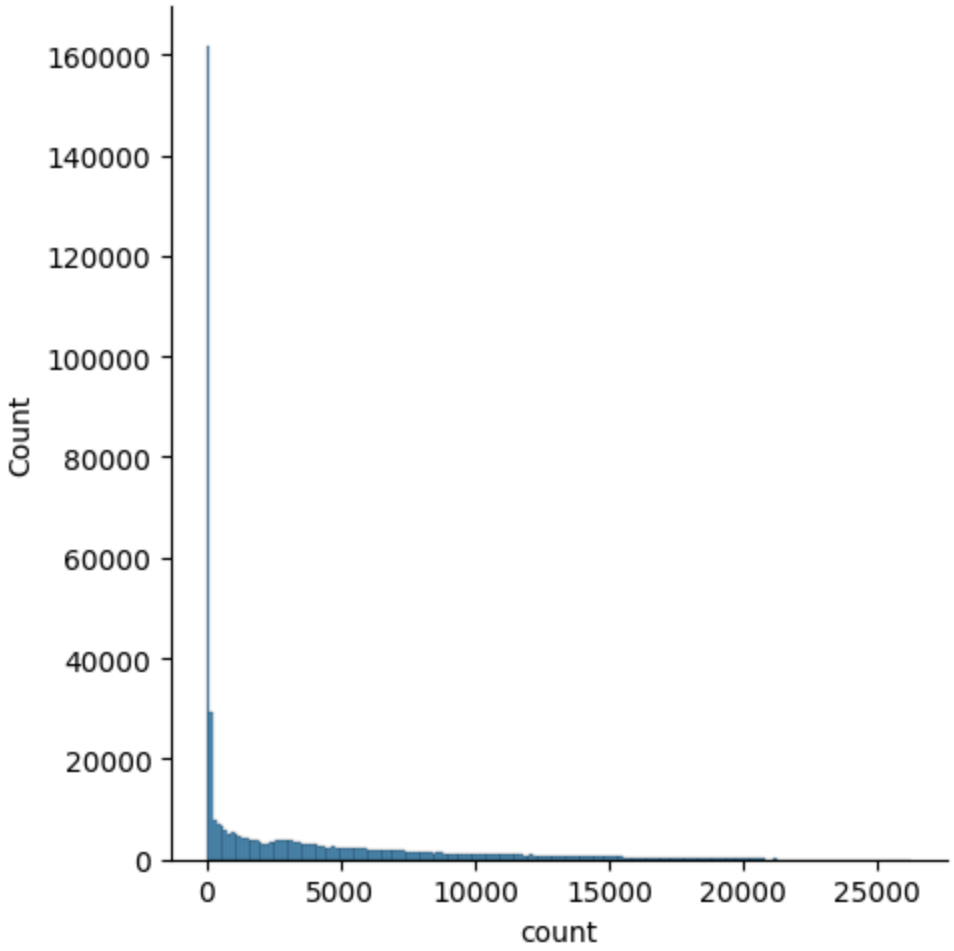
	Day	DoWeek	hour	count	passed	period	data	ServiceID	C
0	12/29/2021	2	17	2974	2974	63.952589	612.081036	2	
1	12/29/2021	2	17	4528	4528	63.000000	960.307862	11	
2	12/29/2021	2	17	19	19	62.000000	566.315790	5	
3	12/29/2021	2	17	1207	1207	63.000000	601.025684	7	
4	12/29/2021	2	18	8450	8450	63.942130	611.024024	2	
...	...	...	...	...	...	...	...	...	
443006	7/24/2023	0	18	1	1	62.000000	624.000000	1	
443007	7/24/2023	0	18	108	108	62.000000	1075.972222	5	
443008	7/24/2023	0	18	5279	5279	63.000000	599.900170	7	
443009	7/24/2023	0	18	2	2	63.000000	1437.500000	0	
443010	7/24/2023	0	18	18	18	0.000000	378255.166700	10	

443011 rows × 9 columns

3.2 Remove outliers

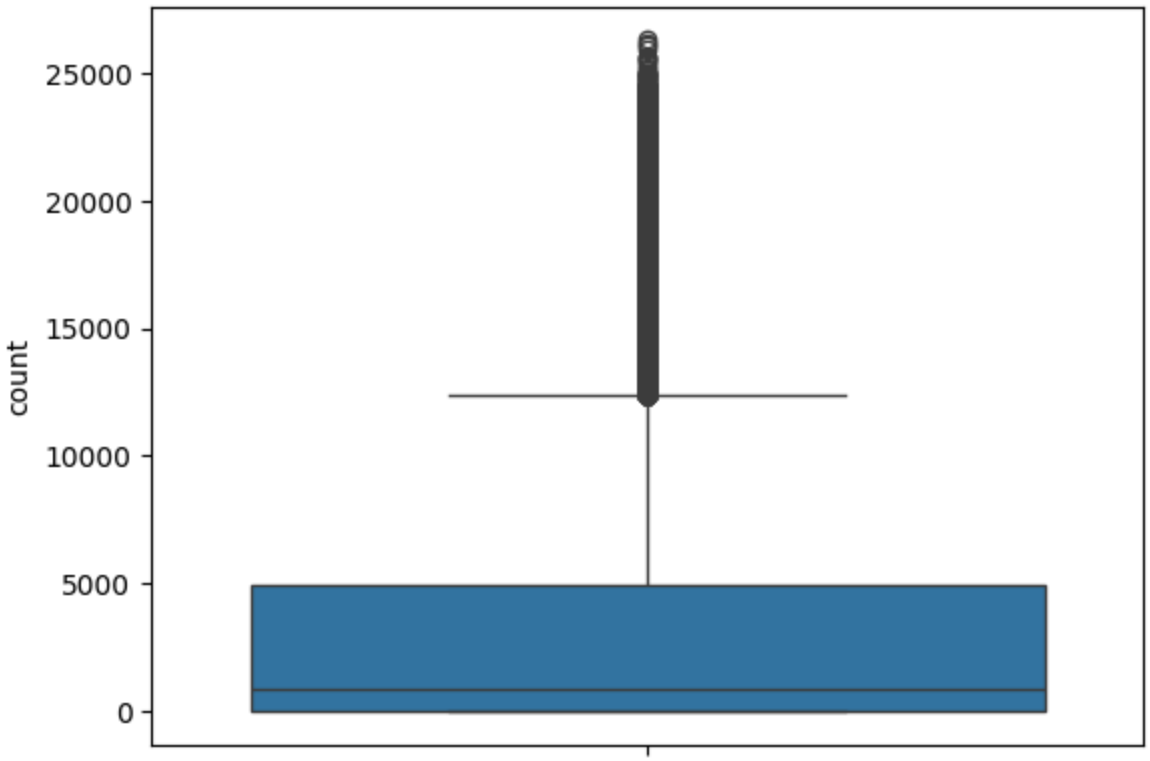
In [10]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.displot(df["count"])
```

Out[10]: <seaborn.axisgrid.FacetGrid at 0x212d99c4c20>



In [11]:
```
#to see outliers clearly
sns.boxplot(df["count"])
```

Out[11]: <Axes: ylabel='count'>



In [12]:
```
#find the limits
upper_limit = df["count"].mean() + 3*df["count"].std()
lower_limit = df["count"].mean() - 3*df["count"].std()
print("upper limit: ", upper_limit)
print("lower limit: ", lower_limit)
```

```
upper limit:  16916.48442636824
lower limit:  -10452.477031518
```

In [13]:
```
df.loc[(df["count"] > upper_limit) | (df["count"] < lower_limit)]
```

Out[13]:

	Day	DoWeek	hour	count	passed	period	data	ServiceID	Ou
154	12/30/2021	3	17	17191	17191	63.000000	1014.325112	11	
241	12/31/2021	4	9	19038	19038	62.999895	995.797353	11	
295	12/31/2021	4	15	19304	19304	62.999896	997.908050	11	
306	12/31/2021	4	16	22546	22546	63.000000	989.394793	11	
317	12/31/2021	4	17	18565	18565	63.000000	986.830434	11	
...	...	...	...	...	...	...	...	...	
442923	7/12/2023	2	17	17091	17091	63.959979	671.547832	2	
442925	7/12/2023	2	17	21684	21684	62.999908	916.707019	11	
442939	7/14/2023	4	17	19416	19416	62.999794	928.368305	11	
442955	7/17/2023	0	17	17514	17514	63.000000	921.919093	11	
442984	7/21/2023	4	17	17297	17297	63.000000	921.184252	11	

8093 rows × 9 columns

In [14]:
```
#Triming - delete the outliers data
old_df = df
df = df.loc[(df["count"] < upper_limit) & (df["count"] > lower_limit)]
```

In [15]:
```
print("Old data", len(old_df))
print("New data", len(df))
```

```
Old data 443011
New data 434918
```

3.3 Balance for train data set

Label for datas

In [16]:
```
import numpy as np
df["Outcome"] = np.where((df["period"] > 63) | (df["passed"] < df["count"]), 1, 0)
df = df.drop("Day", axis=1)
df
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_23516\1763888691.py:2: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df["Outcome"] = np.where((df["period"] > 63) | (df["passed"] < df["count"]), 1, 0)
```

Out[16]:

	DoWeek	hour	count	passed	period	data	ServiceID	Outcome
0	2	17	2974	2974	63.952589	612.081036	2	1
1	2	17	4528	4528	63.000000	960.307862	11	0
2	2	17	19	19	62.000000	566.315790	5	0
3	2	17	1207	1207	63.000000	601.025684	7	0
4	2	18	8450	8450	63.942130	611.024024	2	1
...	...	...	...	...	...	...	...	...
443006	0	18	1	1	62.000000	624.000000	1	0
443007	0	18	108	108	62.000000	1075.972222	5	0
443008	0	18	5279	5279	63.000000	599.900170	7	0
443009	0	18	2	2	63.000000	1437.500000	0	0
443010	0	18	18	18	0.000000	378255.166700	10	0

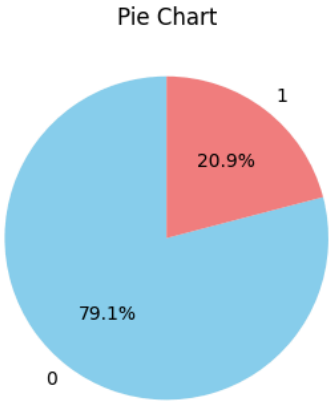
434918 rows × 8 columns

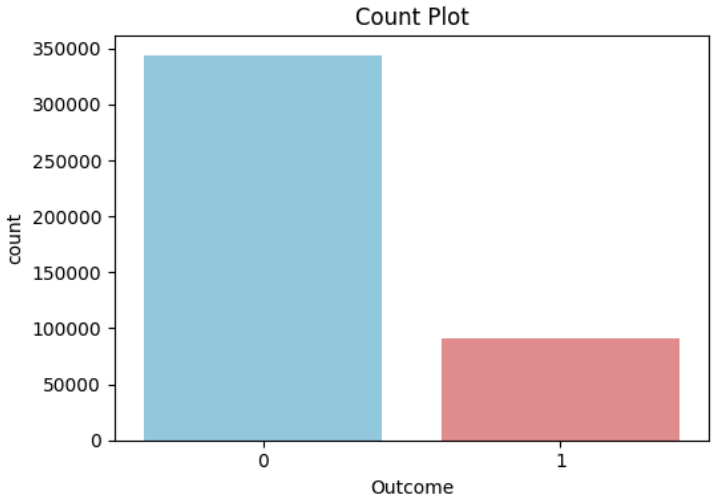
Split for data

In [17]:
```
#Data split
x = df.drop("Outcome", axis= 1)
y = df["Outcome"]
```

In [18]:
```
import matplotlib.pyplot as plt
import seaborn as sns

#Set up the subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize= (10,4) )

#Pie chart for class distribution
pie_colors = ['skyblue', 'lightcoral']
axes[0].pie(df['Outcome'].value_counts(), labels = df['Outcome'].value_counts().ind
axes[0].set_title("Pie Chart")

countplot_colors = sns.color_palette(pie_colors)
sns.countplot(x = 'Outcome', data=df, palette=countplot_colors, ax=axes[1])
axes[1].set_title('Count Plot')

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

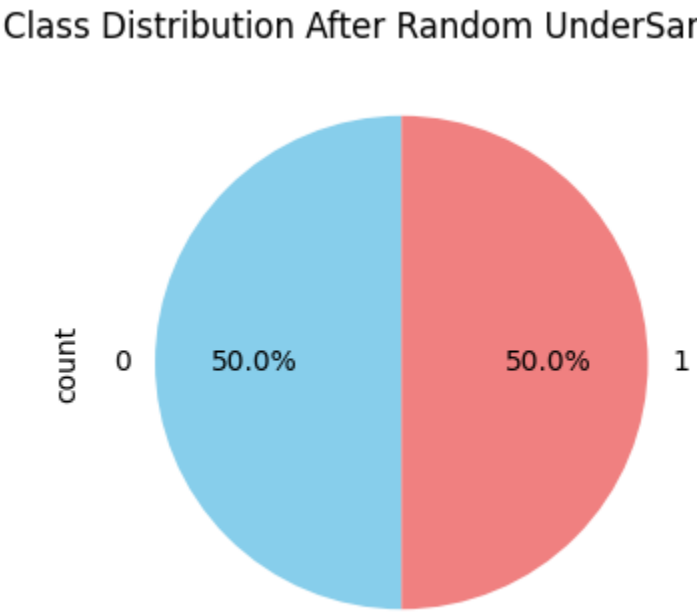
```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_23516\2026622974.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x = 'Outcome', data=df, palette=countplot_colors, ax=axes[1])
```





In [19]:
```
def plot_resampling_results(y_resamples, title):
    plt.figure(figsize= (4,4))
    pd.Series(y_resamples).value_counts().plot.pie(autopct= '%1.1f%%', startangle=
    plt.title(title)
    plt.show()
```

In [20]:
```
from imblearn.under_sampling import NearMiss
nm = NearMiss()
x_res, y_res = nm.fit_resample(x, y)
plot_resampling_results(y_res, "Class Distribution After Random UnderSampling")
```



In [21]:
```
x_res.shape, y_res.shape
```

Out[21]: ((181968, 7), (181968,))

In [22]:
```
from collections import Counter
print("Original dataset shape{}".format(Counter(y)))
```

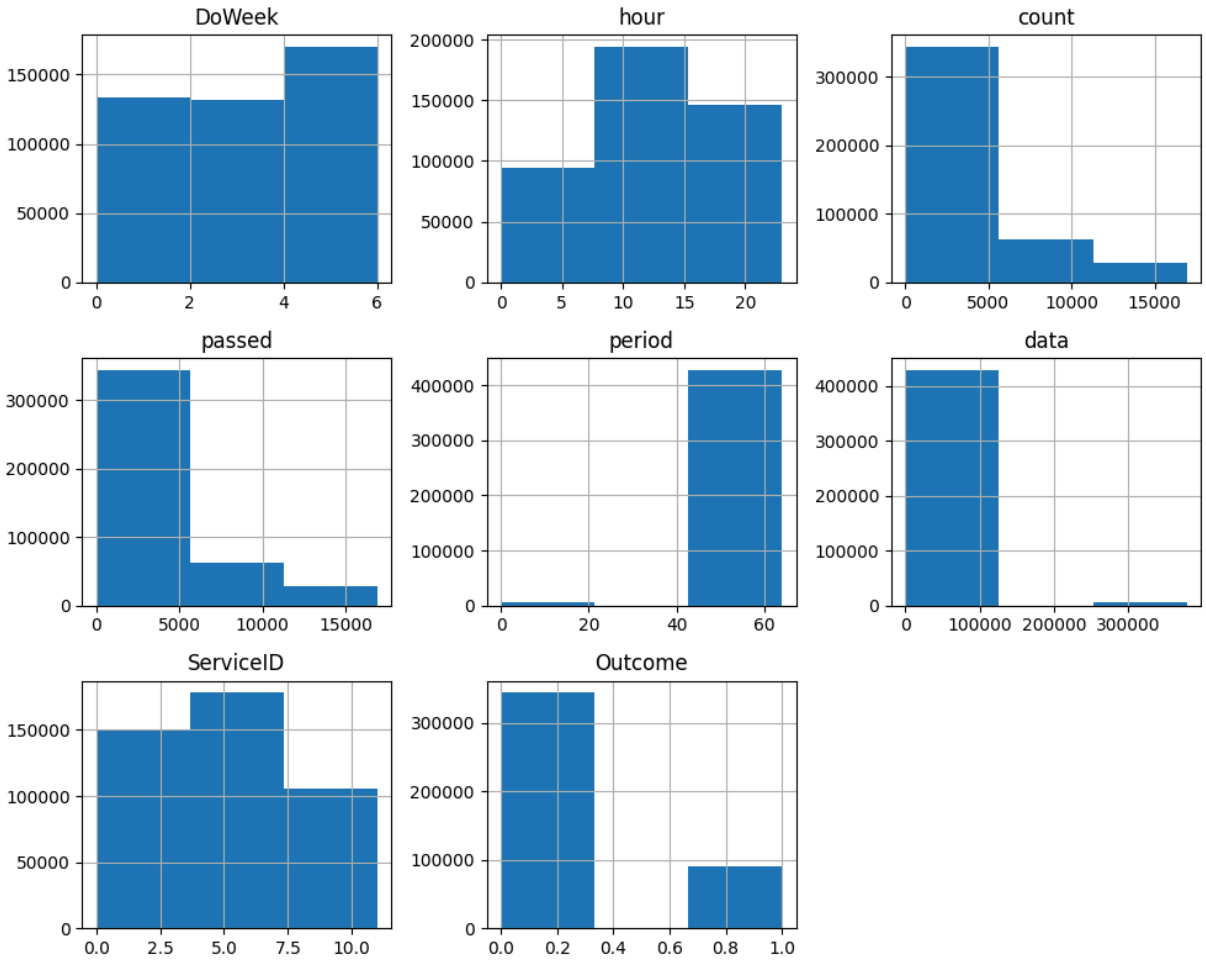
```
print("Resampled dataset shape{}".format(Counter(y_res)))
```

```
Original dataset shapeCounter({0: 343934, 1: 90984})
Resampled dataset shapeCounter({0: 90984, 1: 90984})
```

4. Data visualization

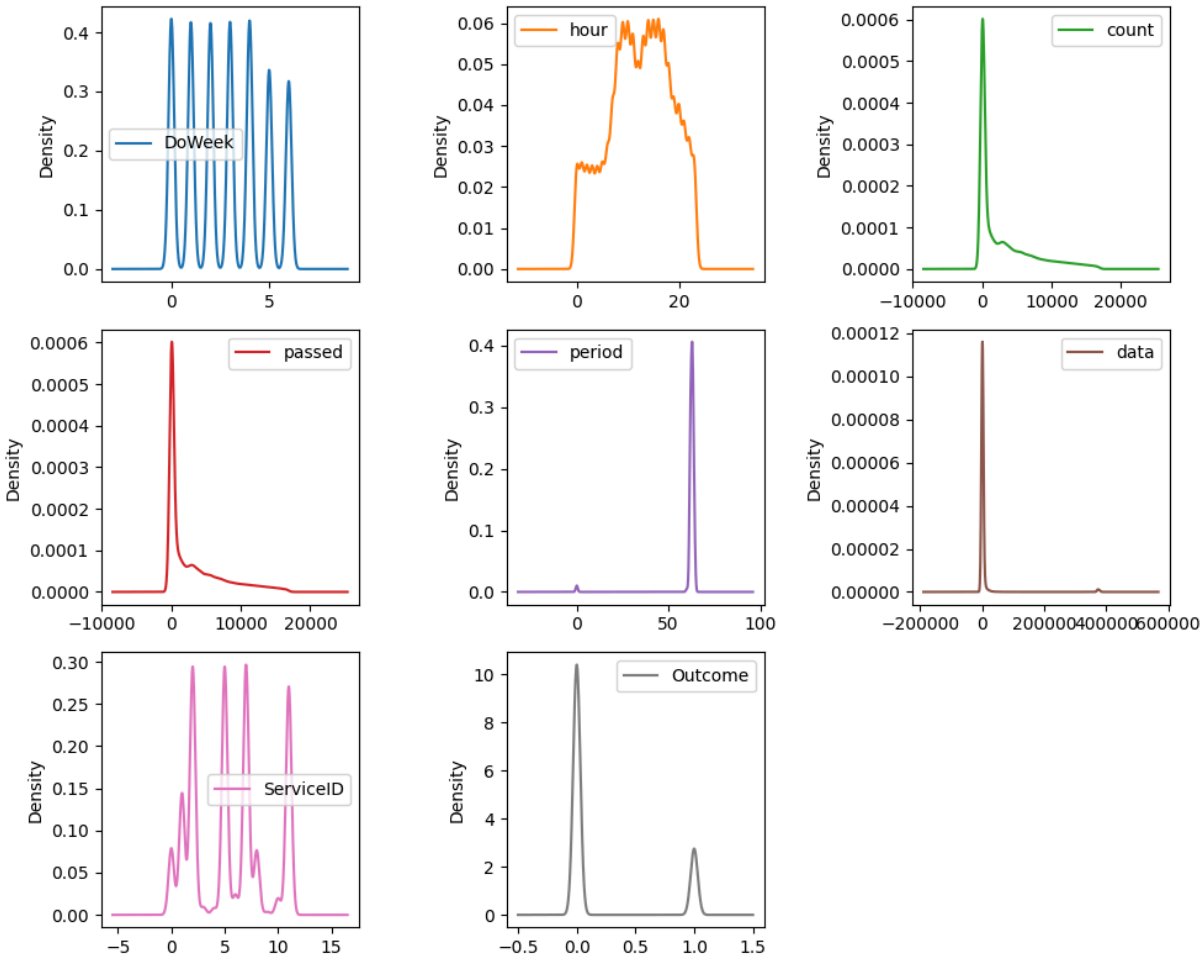
Histogram

In [23]:
```
df.hist(bins=3, figsize=(10, 8), layout=(3, 3))
plt.tight_layout()
plt.show()
```



Density Plot

In [24]:
```
df.plot(kind="density", figsize=(10, 8), subplots=True, layout=(3,3), sharex=False)
plt.tight_layout()
plt.show()
```



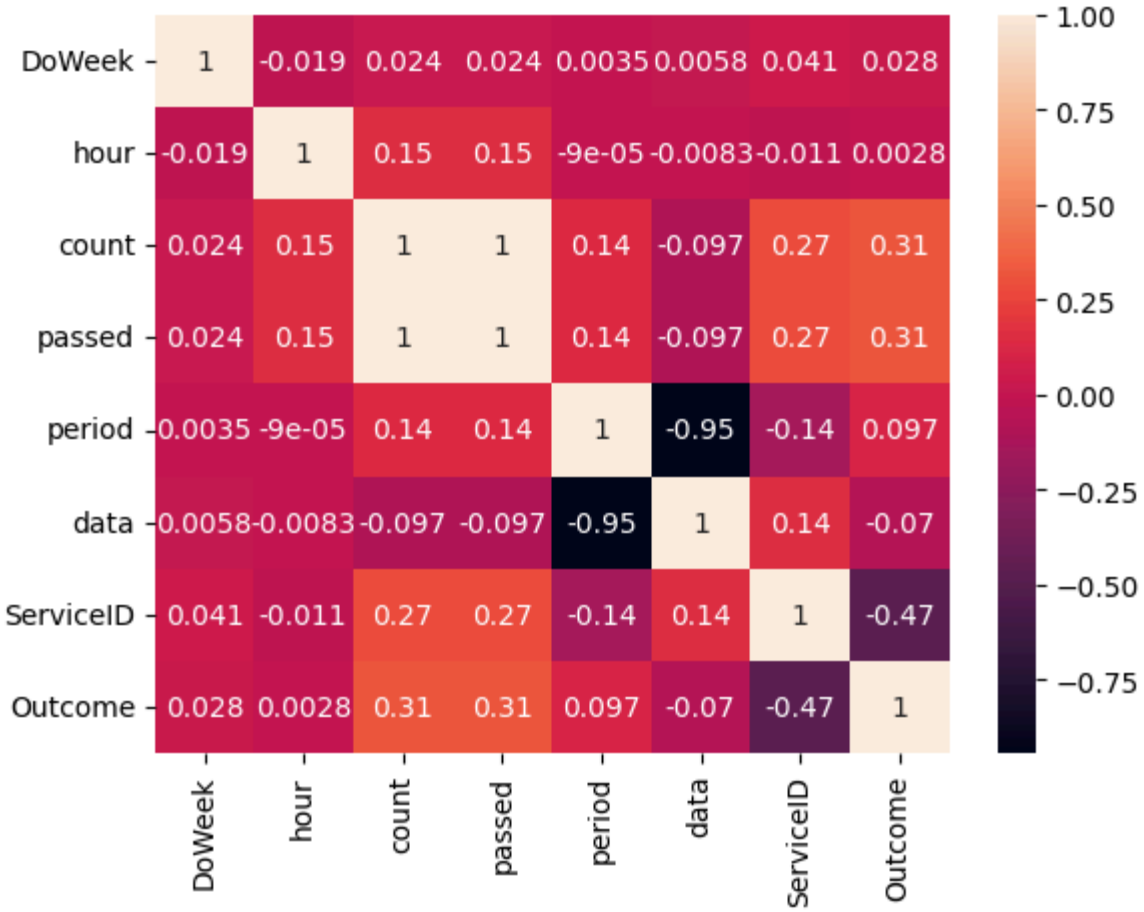
Correlation Matrix Plot

In [25]:
```
df.corr()
```

Out[25]:

	DoWeek	hour	count	passed	period	data	ServiceID	Outc
DoWeek	1.000000	-0.019092	0.023980	0.023966	0.003477	0.005759	0.041427	0.02
hour	-0.019092	1.000000	0.151486	0.151485	-0.000090	-0.008266	-0.011163	0.00
count	0.023980	0.151486	1.000000	0.999993	0.140583	-0.096548	0.273525	0.31
passed	0.023966	0.151485	0.999993	1.000000	0.140608	-0.096540	0.273505	0.31
period	0.003477	-0.000090	0.140583	0.140608	1.000000	-0.945011	-0.141333	0.09
data	0.005759	-0.008266	-0.096548	-0.096540	-0.945011	1.000000	0.143816	-0.07
ServiceID	0.041427	-0.011163	0.273525	0.273505	-0.141333	0.143816	1.000000	-0.47
Outcome	0.027692	0.002834	0.312334	0.312210	0.097487	-0.070267	-0.470000	1.00

In [26]:
```
sns.heatmap(df.corr(), annot=True)
plt.show()
```



4. Model Building

In [27]:
```
from lazypredict.Supervised import LazyClassifier # type: ignore
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report


x_train, x_test, y_train, y_test = train_test_split(x_res, y_res,test_size=.75,rand

clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None)
models,predictions = clf.fit(x_train, x_test, y_train, y_test)
models
```

```
 97%|█████████▋| 30/31 [06:05<00:05,  5.88s/it]
[LightGBM] [Info] Number of positive: 22661, number of negative: 22831
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing wa
s 0.000569 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 45492, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498132 -> initscore=-0.007474
[LightGBM] [Info] Start training from score -0.007474
100%|██████████| 31/31 [06:05<00:00, 11.80s/it]
```

Out[27]:

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
LGBMClassifier	0.98	0.98	0.98	0.98	0.46
XGBClassifier	0.98	0.98	0.98	0.98	0.86
RandomForestClassifier	0.98	0.98	0.98	0.98	4.15
BaggingClassifier	0.98	0.98	0.98	0.98	1.10
AdaBoostClassifier	0.98	0.98	0.98	0.98	1.89
ExtraTreesClassifier	0.98	0.98	0.98	0.98	2.08
SVC	0.98	0.98	0.98	0.98	20.52
KNeighborsClassifier	0.98	0.98	0.98	0.98	7.68
DecisionTreeClassifier	0.97	0.97	0.97	0.97	0.21
ExtraTreeClassifier	0.97	0.97	0.97	0.97	0.08
BernoulliNB	0.96	0.96	0.96	0.96	0.07
SGDClassifier	0.93	0.93	0.93	0.93	0.12
NuSVC	0.93	0.93	0.93	0.93	200.54
Perceptron	0.93	0.93	0.93	0.93	0.09
LinearDiscriminantAnalysis	0.92	0.92	0.92	0.92	0.28
RidgeClassifierCV	0.92	0.92	0.92	0.92	0.08
LinearSVC	0.92	0.92	0.92	0.92	0.15
CalibratedClassifierCV	0.92	0.92	0.92	0.92	0.32
RidgeClassifier	0.92	0.92	0.92	0.92	0.08
LogisticRegression	0.92	0.92	0.92	0.92	0.15
GaussianNB	0.91	0.91	0.91	0.91	0.08
NearestCentroid	0.90	0.90	0.90	0.90	0.12
PassiveAggressiveClassifier	0.80	0.80	0.80	0.80	0.10
QuadraticDiscriminantAnalysis	0.69	0.69	0.69	0.65	0.10
DummyClassifier	0.50	0.50	0.50	0.33	0.05

In [ ]:

Choose LGBMClassifier Model based on EXECUTION TIME and F1 SCORE

In [28]:
```
from sklearn.preprocessing import StandardScaler
from lightgbm import LGBMClassifier


scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

lgb_clf = LGBMClassifier(random_state=101)
lgb_clf.fit(x_train, y_train)
y_predict = lgb_clf.predict(x_test)
lgb_clf.score(x_test, y_test)
```

```
[LightGBM] [Info] Number of positive: 22661, number of negative: 22831
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing wa
s 0.000181 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 45492, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498132 -> initscore=-0.007474
[LightGBM] [Info] Start training from score -0.007474
```

Out[28]: 0.98292007385914

In [29]:
```
for i, j in zip(y_predict[:10], y_test.values[:10]):
    print("Prediction: {}. Actual value: {}".format(i, j))
```

```
Prediction: 1. Actual value: 1
Prediction: 1. Actual value: 1
Prediction: 0. Actual value: 1
Prediction: 0. Actual value: 0
Prediction: 0. Actual value: 0
Prediction: 1. Actual value: 1
Prediction: 1. Actual value: 1
Prediction: 0. Actual value: 0
Prediction: 0. Actual value: 0
Prediction: 0. Actual value: 0
```

Model report

In [30]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     68153
           1       1.00      0.97      0.98     68323

    accuracy                           0.98    136476
   macro avg       0.98      0.98      0.98    136476
weighted avg       0.98      0.98      0.98    136476
```

5. Parameter Adjustment

In [31]:
```
from sklearn.model_selection import GridSearchCV

params = {
    "boosting_type": ["gbdt", "dart", "rf"],
    "num_leaves": list(range(20,40)),
    "min_child_samples": list(range(15,25))
}
grid_search = GridSearchCV(estimator=LGBMClassifier(random_state=101), param_grid=p
grid_search.fit(x_train, y_train)
print(grid_search.best_params_)
print(grid_search.best_estimator_)
print(grid_search.best_score_)
```

```
[LightGBM] [Info] Number of positive: 18129, number of negative: 18264
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000571 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 36393, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498145 -> initscore=-0.007419
[LightGBM] [Info] Start training from score -0.007419
[LightGBM] [Info] Number of positive: 18128, number of negative: 18265
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing wa
s 0.000166 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 36393, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498118 -> initscore=-0.007529
[LightGBM] [Info] Start training from score -0.007529
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000485 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 36394, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498132 -> initscore=-0.007474
[LightGBM] [Info] Start training from score -0.007474
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000524 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 36394, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498132 -> initscore=-0.007474
[LightGBM] [Info] Start training from score -0.007474
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000773 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 36394, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498132 -> initscore=-0.007474
[LightGBM] [Info] Start training from score -0.007474
[LightGBM] [Info] Number of positive: 18129, number of negative: 18264
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing wa
s 0.000151 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 36393, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498145 -> initscore=-0.007419
[LightGBM] [Info] Start training from score -0.007419
```

```
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18264
[LightGBM] [Info] Number of positive: 18128, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18264
[LightGBM] [Info] Number of positive: 18128, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18264
[LightGBM] [Info] Number of positive: 18128, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18264
[LightGBM] [Info] Number of positive: 18128, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 18129, number of negative: 18265
[LightGBM] [Info] Number of positive: 22661, number of negative: 22831
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing wa
s 0.000210 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 45492, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498132 -> initscore=-0.007474
[LightGBM] [Info] Start training from score -0.007474
{'boosting_type': 'gbdt', 'min_child_samples': 20, 'num_leaves': 29}
LGBMClassifier(num_leaves=29, random_state=101)
0.983689483824773
```

Choose 'boosting_type': 'gbdt', 'min_child_samples': 20, 'num_leaves': 29

In [32]:
```
lgb_clf = LGBMClassifier(boosting_type="gbdt", num_leaves=29, min_child_samples=20,
lgb_clf.fit(x_train, y_train)
y_predict = lgb_clf.predict(x_test)
print(classification_report(y_test, y_predict))
print(x_train[1])
```

```
[LightGBM] [Info] Number of positive: 22661, number of negative: 22831
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing wa
s 0.000175 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1065
[LightGBM] [Info] Number of data points in the train set: 45492, number of used feat
ures: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498132 -> initscore=-0.007474
[LightGBM] [Info] Start training from score -0.007474
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     68153
           1       1.00      0.97      0.98     68323

    accuracy                           0.98    136476
   macro avg       0.98      0.98      0.98    136476
weighted avg       0.98      0.98      0.98    136476

[-1.47248149  1.37815026  0.3802959   0.38045748 -0.02151186 -0.10159558
  1.03766159]
```

6. Model deployment

In [33]:
```
def predict_order(DoWeek, hour, count, passed, period, data, ServiceId):
    x= np.zeros(7)
    x[0] = DoWeek
    x[1] = hour
    x[2] = count
    x[3] = passed
    x[4] = period
    x[5] = data
    x[6] = ServiceId
    return int(lgb_clf.predict([x])[0])
```

In [35]:
```
predicted_label = predict_order(2, 17, 1000, 99, 60, 600, 6)
print(predicted_label)
if predicted_label == 0:
    print("Đơn hàng  o xảy ra sự cố")
else:
    print("Đơn hàng  xảy ra sự cố")
```

```
1
Đơn hàng  xảy ra sự cố
```