Jupyter Notebook Execution Report

Name: Your Name

Date: December 09, 2025

Cell 1: ■ Code

```
import pandas as pd
import sqlite3
import numpy as np
```

Cell 2: ■ Code

```
conn = sqlite3.connect("../Nexus.db")
#notifications = pd.read_sql_query("SELECT * FROM notifications", conn)
notifications = pd.read_csv("../D_download_stock_price/car.csv", encoding="utf-8")
#stock_prices = pd.read_sql_query("SELECT * FROM stock_prices", conn)
#osbx = pd.read_sql_query("SELECT * FROM osbx", conn)
#insiders_transactions = pd.read_sql_query("SELECT * FROM insider_transactions",
conn)
transactions = pd.read_csv("transactions_agg.csv", encoding="utf-8")
conn.close()
```

Cell 3: ■ Code

```
notifications.head()
```

Output:

```
       id            company_name  ...  event_date    car_42
0  15939  2020 Bulkers Ltd..json  ...  2019-08-15  0.009525
1  15938  2020 Bulkers Ltd..json  ...  2019-08-22  0.033167
2  15937  2020 Bulkers Ltd..json  ...  2019-08-23  0.026759
3  15936  2020 Bulkers Ltd..json  ...  2019-11-27 -0.050793
4  15935  2020 Bulkers Ltd..json  ...  2019-11-28 -0.103633

[5 rows x 10 columns]
```

Cell 4: ■ Code

```
#print((notifications["car_10"] &gt; 0).sum())
#print((notifications["car_20"] &gt; 0).sum())
print((notifications["car_42"] &gt; 0).sum())
```

Output:

```
4224
```

Cell 5: ■ Code

```
targets = notifications[["id", "car_42"]]
df_model = transactions.merge(targets, left_on="notification_id", right_on="id",
how="inner")
```

Cell 6: ■ Code

```
print(len(df_model))
print((df_model["car_42"] == 0.0000).sum()) # might need to check why so many have
0?, dont have time just droping them...
df_model = df_model[df_model["car_42"] != 0.0]
```

Output:

```
6786
2967
6786
2967
```

Cell 7: ■ Code

```
len(df_model)
```

Output:

```
3819
```

Cell 8: ■ Code

```
df_model = df_model.drop(columns=["id", "notification_id"])
```

```
df_model
```

Output:

```
      salary_related         transaction_type_mapped  ...  n_people    car_42
12                 0         Disposal/Sale of Shares  ...         2 -0.120709
135                0  Acquisition/Purchase of Shares  ...         1 -0.014134
136                0  Acquisition/Purchase of Shares  ...         1 -0.030012
137                0  Acquisition/Purchase of Shares  ...         1 -0.030012
138                0  Acquisition/Purchase of Shares  ...         1 -0.030012
...              ...                             ...  ...       ...       ...
6738               0  Acquisition/Purchase of Shares  ...         1       NaN
6771               0  Acquisition/Purchase of Shares  ...         1 -0.196316
6772               0  Acquisition/Purchase of Shares  ...         1 -0.273016
6773               0  Acquisition/Purchase of Shares  ...         1 -0.277037
6774               0  Acquisition/Purchase of Shares  ...         1 -0.306110

[3819 rows x 6 columns]
```

Cell 9: ■ Code

```
df_model["car_3class"] = np.select(
[
df_model["car_42"] &lt; -0.0, # Class 0
#df_model["car_42"].between(-0.02, 0.02), # Originaly beffor droping 0 values ...
df_model["car_42"] &gt; 0.0, # Class 1
],
[0, 1]
)
```

Cell 10: ■ Code

```
df_model["car_3class"].value_counts()
```

Output:

```
car_3class
0    2012
1    1807
Name: count, dtype: int64
```

Cell 11: ■ Code

```
df_model["salary_related"].value_counts()
```

Output:

```
salary_related
0    2649
1    1170
Name: count, dtype: int64
```

Cell 12: ■ Code

```
df_model["transaction_type_mapped"].value_counts()
```

Output:

```
transaction_type_mapped
Acquisition/Purchase of Shares           2705
Disposal/Sale of Shares                   637
Group Not Found                           140
Grant/Award of Rights/Instruments         124
Exercise/Settlement leading to Shares      72
Transfer (Internal/Other)                  45
Administrative/Other/No Transaction        39
Lending/Pledging/Agreement                 16
Share Buyback/Repurchase                   14
Acquisition of Rights/Instruments          14
Exercise and Sale (Net Disposal)            4
Disposal of Rights/Instruments              3
Cash Settlement of Rights/Options           2
Transfer (Outflow/Internal)                 2
Transfer (Rights)                           1
Unclassified/Needs Review                   1
Name: count, dtype: int64
```

Cell 13: ■ Code

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
```

```
X = df_model[[
"salary_related",
"transaction_type_mapped",
"percentage_change_in_holding",
"volume",
"n_people"
]]
```

```
y = df_model["car_3class"]
```

```
# Column lists
categorical = ["transaction_type_mapped"]
numeric = [
"salary_related",
"percentage_change_in_holding",
"volume",
"n_people"
]
```

```
# one-hot encode the category, keeping numeric features as is
preprocess = ColumnTransformer(
transformers=[
("cat", OneHotEncoder(sparse_output=False, handle_unknown="ignore"), categorical),
("num", "passthrough", numeric)
]
)
```

```
# RandomForest model
rf = RandomForestClassifier(
n_estimators=300,
class_weight="balanced",
```

```
max_depth=None,
min_samples_split=15,
random_state=42
)
```

```
# Full pipeline
model = Pipeline(steps=[
("preprocess", preprocess),
("rf", rf)
])
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=42
)
```

```
# Fit
model.fit(X_train, y_train)
```

Output:

```
Pipeline(steps=[('preprocess',
                 ColumnTransformer(transformers=[('cat',
                                                  OneHotEncoder(handle_unknown='ignore',
                                                                sparse_output=False),
                                                  ['transaction_type_mapped']),
                                                 ('num', 'passthrough',
                                                  ['salary_related',
                                                   'percentage_change_in_holding',
                                                   'volume', 'n_people'])])),
                ('rf',
                 RandomForestClassifier(class_weight='balanced',
                                        min_samples_split=15, n_estimators=300,
                                        random_state=42))])
```

Cell 14: ■ Code

```
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test, model.predict(X_test)))
```

Output:

```
              precision    recall  f1-score   support

           0       0.60      0.62      0.61       594
           1       0.57      0.55      0.56       552

    accuracy                           0.59      1146
   macro avg       0.59      0.59      0.59      1146
weighted avg       0.59      0.59      0.59      1146
```

Cell 15: ■ Code

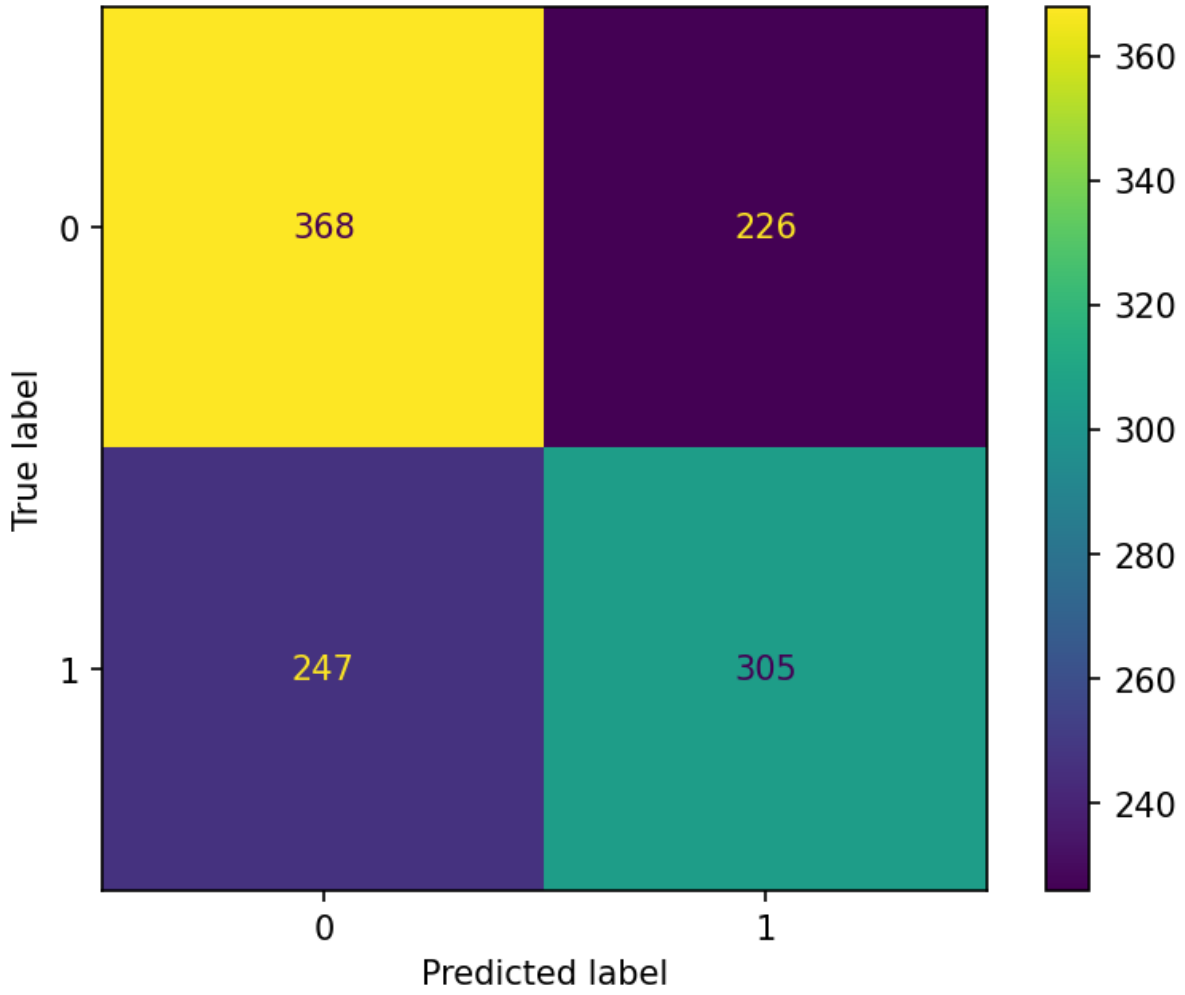
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
```

```
y_pred = model.predict(X_test)
```

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

Output:

```
[STDERR]
&lt;string&gt;:1: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
```



Cell 16: ■ Code

```
model.feature_impotance_
```

Error:

```
Traceback (most recent call last):
  File "/home/imre/.vscode-server/extensions/ganeshkumbhar.nb2pdf-1.1.9/scripts/nb2pdf.py", line
    result = eval(lines[-1], glb)
  File "&lt;string&gt;", line 1, in &lt;module&gt;
AttributeError: 'Pipeline' object has no attribute 'feature_impotance_'
```

Cell 17: ■ Code

```
df_model = df_model[(df_model["salary_related"] == 0)]

df_model = df_model[(df_model["transaction_type_mapped"] == "Acquisition/Purchase
of Shares")]

df_model = df_model[(df_model["percentage_change_in_holding"] &gt;= 0.5)]
```

```
df_model = df_model[(df_model["volume"] &gt;= 950000)]
df_model = df_model[(df_model["n_people"] == 1)]
```

Cell 18: ■ Code

```
df_model["car_3class"].value_counts()
```

Output:

```
car_3class
0    98
1    80
Name: count, dtype: int64
```

Cell 19: ■ Code