Python meeting

06: Python meeting 7: IsolationForest

27th May 2026

The Python Meeting will start soon

COURSE PYTHON SCHEDULE

01: 20250114, 9am: Initiation – Case-study: P02_15: Above average prices

- Set-up: Python, Visual Studio Code
- Project creation: Folder, .py, Virtual Environment, CMD
- Libraries: Import, requirements file
- Naming conventions
- Basic objects: Dataframe (table), Series (column), Variable, Function, Group_by
- Basic functions: Import, Rename, Filter, Join, Create columns, Group, Aggregate, Export

02: 20250121,10am: Melt, split, for – Case-studies: D01: Trial balance, P01_19: Supplier debtors, O01_19: Customer creditors

- Using melt, split
- For loop

03: 20250128, 9am: Web-download – Case-study: P01_10: Suppliers in OFAC

- Request for downloading sanctions
- IO for encoding
- Regex for comparison
- Split, explode for row generation
- Levenshtein distance scoring

04: 20250204, 9am: JSON, expressions, rolling Window – Case-study: P02_19: Split POs

- Import JSON, Expression object for adding labels,
- cum_sum() for rolling window

05: 20250211, 9AM: zip, dynamic fields – Case-study: P02_03: PO whilst blocked

- Zip to group lists together
- Using dynamic fields in a for loop

09:00 – 09:05

COURSE PYTHON SCHEDULE

06: 20250218, 10am: Cumulative sum, dates – Case-study: P02_18: PO slow rotation

- cum_sum for calculating future or past inventory movements

07: 20250225, 9am: Isolation forest for unusual transactions – Case-study: Unusual bank transfers

- Using isolation forest library to score for unusual transactions

08: 20250304: 9am: SpaCy – Case-study: P01_11/ O01_11: Suppliers/ customers that are people

- Using SpaCy NLP object to categorize names

09: 20250311: 9am: Contract review – Case-study: Scoring of contracts

- Using Gemini model to check for paragraphs of similar meaning

10: 20250318: 10am: Open AI API – Case-study: Generating audit reports

- Using API to OpenAI to generate text for audit reports

11: 20250325: 9am: Regex and Sklearn matching – Case-study: HR03_12/ H403_13: Unusual T&E

- Using Regex and Sklearn to check for unexpected travel and expenses

12: 20250401: 9am: OCR: Image recognition – Case-study: HR03_14: Expenses for others

- Using OCR python library to read travel and expense receipts

09:00 – 09:05

TODAY'S SCHEDULE

01

Case-study

→ Isolation forest for payment transactions: main concept

02

Python script overview

How to run the isolation forest model and how to tweak the values and review the result

03

Python script step-by-step

Calculation of mean, ratios, log values, rank, percentile, one-hot, hash and isolation forest

09:00 – 09:05

01

Case study: Isolation forest: main concepts

09:00 – 09:05

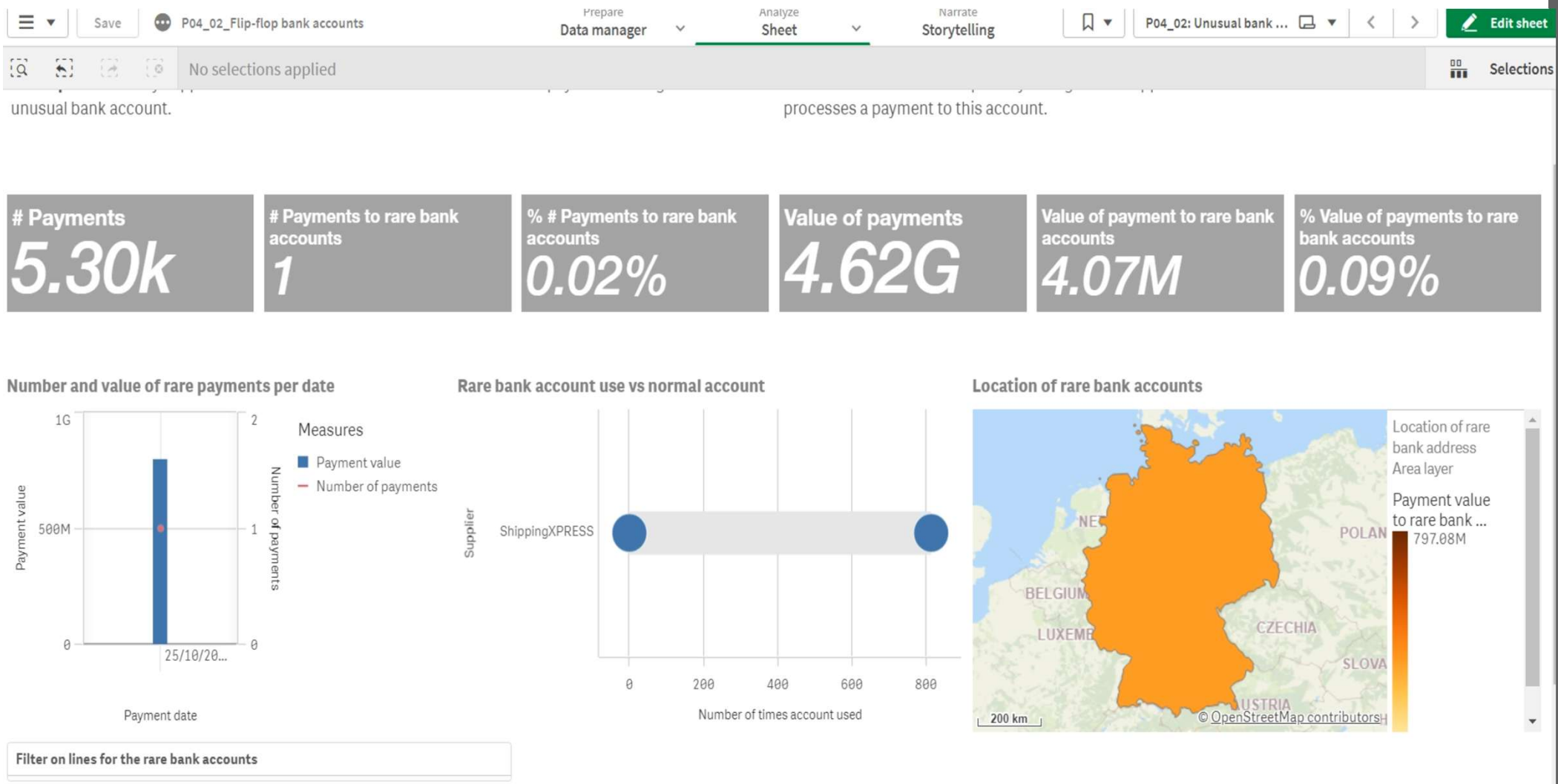
Already using Artificial Intelligence in audit?

20260527_01_AlreadyUsingAIModels?

1. **Are you already using AI models?** (Multiple choice)

- [ ] ChatGPT API
- [ ] Gemini models
- [ ] Amazon bedrock models
- [ ] Python libraries such as IsolationForest
- [ ] Other NLP libraries

The simple test – P04_02 – unusual bank account for a third party



Country of bank account different to 3rd party

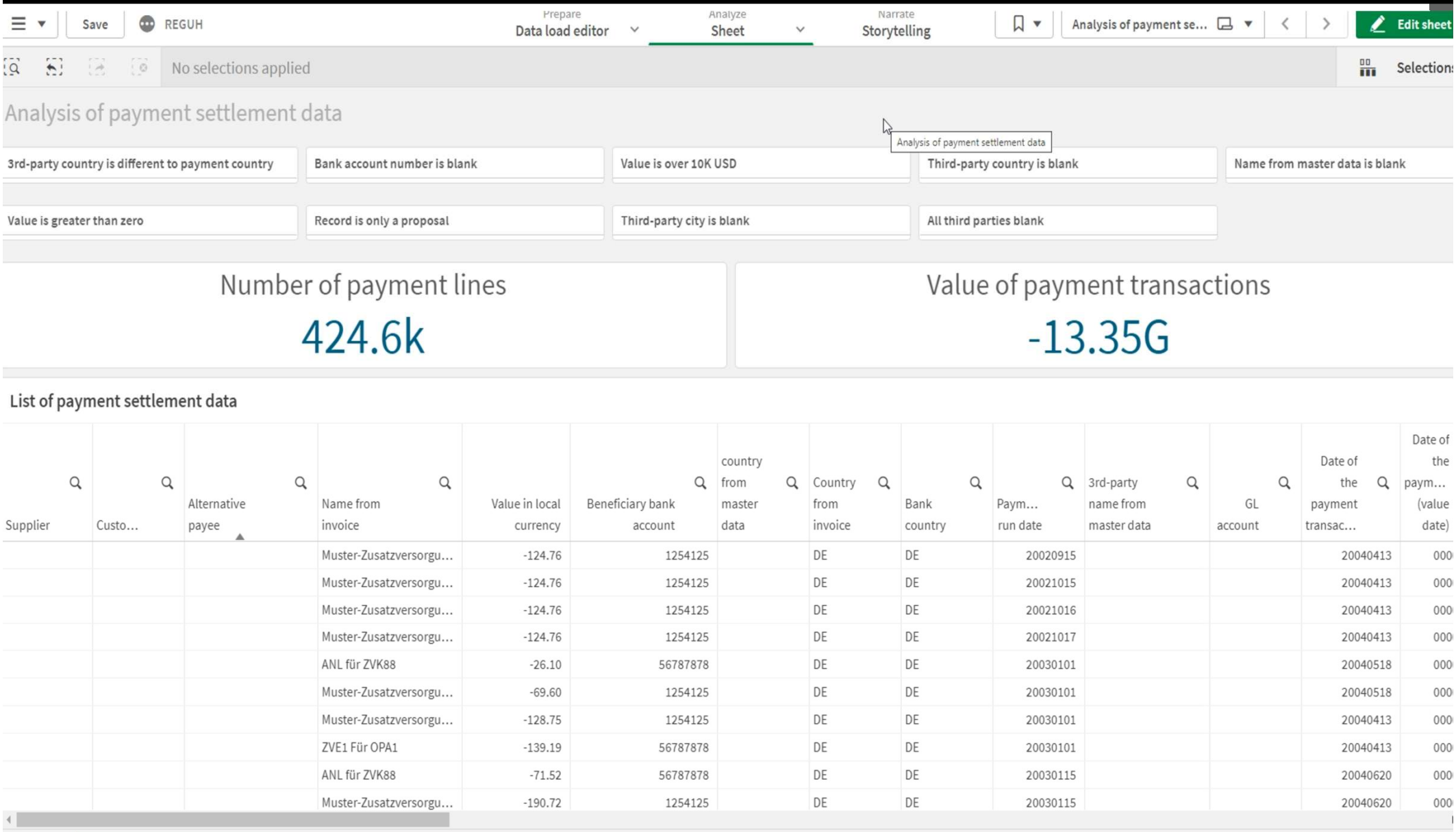
- We could also check the country of the bank account and see if it is ever different to that of the third-party.
- And we could do lots of other simple tests to better understand our file.

Number of payment lines	Value of payment transactions
434	-92.08M

List of payment settlement data

Supplier	Custo...	Alternative payee	Name from invoice	Value in local currency	Beneficiary bank account	Country from master data	Country from invoice	Bank country	Paym... run date	3rd-party name from master data	GL account
VENDOR01			Vendor 01 GB	-10,123,456.00	987654	GB	GB	DE	20081031	Vendor 01 GB	
VENDOR01			Vendor 01 GB	-10,123,456.00	987654	GB	GB	DE	20081031	Vendor 01 GB	
0000001011			SKF Americas	-8,297,574.95	9998233	US	US	DE	20030102	SKF Americas	
0000001011			SKF Americas	-8,297,574.95	9998233	US	US	DE	20030102	SKF Americas	
VENDOR01			Vendor 01 GB	-4,000,000.00	987654	GB	GB	DE	20081031	Vendor 01 GB	

Quick review of the REGUH file



Supplier	Custo...	Alternative payee	Name from invoice	Value in local currency	Beneficiary bank account	country from master data	Country from invoice	Bank country	Paym... run date	3rd-party name from master data	GL account	Date of the payment transac...	Date of the paym... (value date)
			Muster-Zusatzversorgu...	-124.76	1254125		DE	DE	20020915			20040413	000
			Muster-Zusatzversorgu...	-124.76	1254125		DE	DE	20021015			20040413	000
			Muster-Zusatzversorgu...	-124.76	1254125		DE	DE	20021016			20040413	000
			Muster-Zusatzversorgu...	-124.76	1254125		DE	DE	20021017			20040413	000
			ANL für ZVK88	-26.10	56787878		DE	DE	20030101			20040518	000
			Muster-Zusatzversorgu...	-69.60	1254125		DE	DE	20030101			20040518	000
			Muster-Zusatzversorgu...	-128.75	1254125		DE	DE	20030101			20040413	000
			ZVE1 Für OPA1	-139.19	56787878		DE	DE	20030101			20040413	000
			ANL für ZVK88	-71.52	56787878		DE	DE	20030115			20040620	000
			Muster-Zusatzversorgu...	-190.72	1254125		DE	DE	20030115			20040620	000

Add the user

- It would be great to include the user in our analysis as well because the user might also be unusual. However, we don't have the user field in our REGUH table. This is how we can add the user:

REGUH table: settlements data – interesting fields for

Field name	Description
ZBUKR	Company making the payment
ZALDT	Posting date of the payment
VBLNR	Document number of the payment

BKPF table: general ledger header

Field name	Description
BUKRS	Company code
BUDAT	Posting date
BELNR	Accounting document number
USNAM	User name

REGUH with user name

Field name	Description
ZBUKR	Company making the payment
ZALDT	Posting date of the payment
VBLNR	Document number of the payment
USNAM	User name

Different libraries for specific questions

Isolation forest

« If the number of questions we have to ask to isolate the record is low, then the record has a rare set of values »

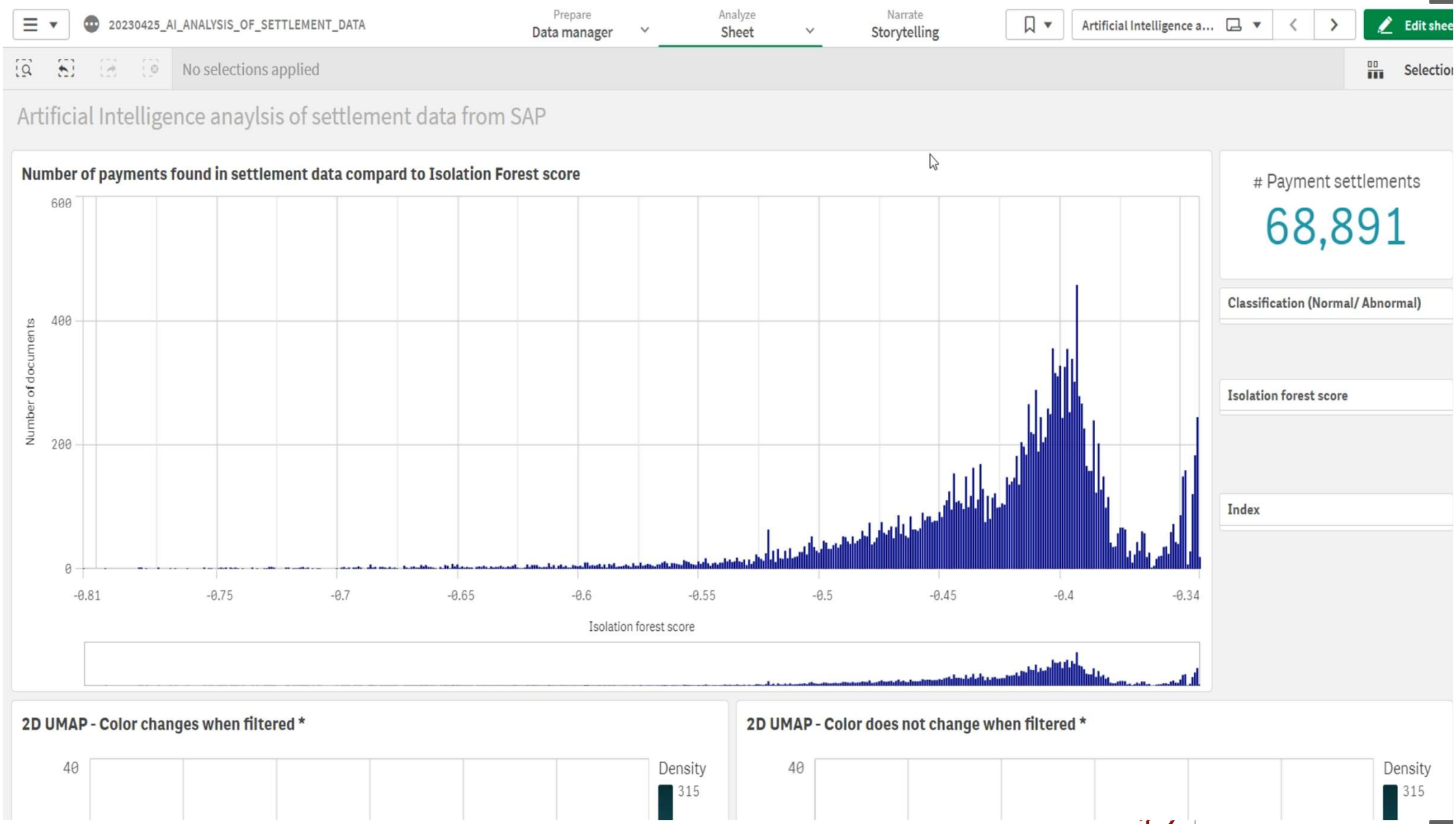
« How can we see records that have a similar set of values to a pre-identified exception? »

UMAP

SHAP

« Why did the record come out as having a rare score in the isolation forest? »

Isolation forest in action!



02

Overview of our python script

Isolation forest script

09:30 – 09:45

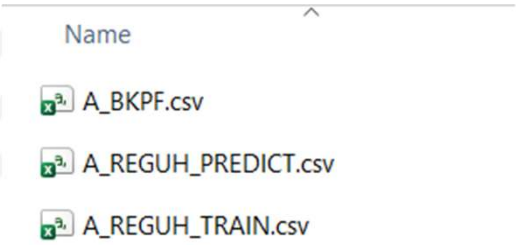
How to use the script

Normally, we train on the normal data and predict on the new data sets….

We can run either the training or the prediction

```
if ZV_ST_TRAIN_OR_PREDICT == 'Train':
    FC_CREATE_TRAIN_ISF(ZV_DF_REGUH)

elif ZV_ST_TRAIN_OR_PREDICT == 'Predict':
    FC_PREDICT_USING_ISF(ZV_DF_REGUH)
```

Change the file name from PREDICT or train to import the one that you updated:



The training will create a model that we will save and the prediction will use the model

We can train on one data set and predict on the other to see the score

09:30 – 09:45

Variables: How to run the program

- Create the virtual environment, activate it and run the requirements.

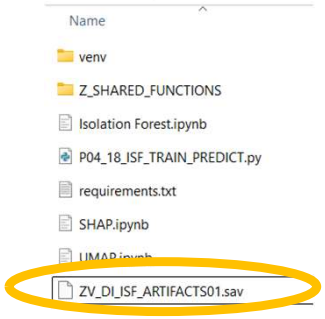
```
requirements.txt
1  polars==1.37.1
2  pandas==2.3.3
3  pyarrow==23.0.0
4  openpyxl==3.1.5
5  numpy==2.4.2
6  scikit-learn==1.8.0
7  pyodbc==5.3.0
8
```

- Update the path variables

- Train the model on the train data set:
  - Write Train for ZV_ST_TRAIN_OR_PREDICT
  - Put A_REGUH_TRAIN as the source file.
  - Run the program: The program will create an Isolation Forest Model

```
20
21  # 3/ Set variables
22  ZV_ST_SOURCES_FOLDER = 'C:/300FMASTERCLASS/06_PYTHON_
23  ZV_04_C_PHASE_FOLDER = 'C:/300FMASTERCLASS/06_PYTHON_
24  ZV_ST_TRAIN_OR_PREDICT = 'Train' # Write Train to tra
25  ZV_ST_NAME_REGUH_FILE = 'A_REGUH_TRAIN.csv' # Note: f
26  ZV_ST_NAME_BKPF_FILE = 'A_BKPF.csv'
27  ZV_ST_NAME_RESULTS_FILE = 'ZV_DF_REGUH_ISF_UPDATED_10
```

  - A .sav file will appear in your programs folder:



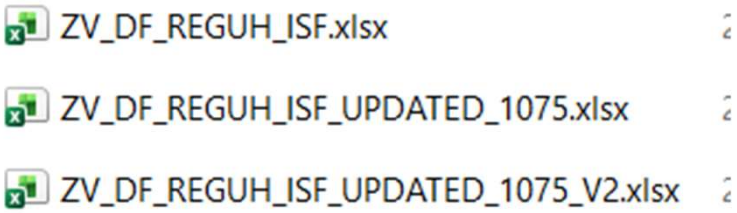
09:30 – 09:45

Variables: How to run the program

- Once you have trained the model – change the variables to Predict and A_REGUH_PREDICT.csv as the source file.
- We can change values in the PREDICT file – in order to see if this changes our score.
- However, if our data set is small, then our changes in the predict file may not make the results change too much… this works better on larger data sets.
- Once we have made our changes, ensure that the source file and the results file are closed, and then run the program again.
- The program will take a few minutes to run and then we will get the results file.

```
# 3/ Set variables
ZV_ST_SOURCES_FOLDER = 'C:/300FMASTERCLASS/06_PYTHON_MEETING_M
ZV_04_C_PHASE_FOLDER = 'C:/300FMASTERCLASS/06_PYTHON_MEETING_M
ZV_ST_TRAIN_OR_PREDICT = 'Predict' # Write Train to train or P
ZV_ST_NAME_REGUH_FILE = 'A_REGUH_PREDICT.csv' # Note: file nam
ZV_ST_NAME_BKPF_FILE = 'A_BKPF.csv'
ZV_ST_NAME_RESULTS_FILE = 'ZV_DF_REGUH_ISF_UPDATED_1075_V2.xls
```

We can experiment with different values in the source file, different isolation forest parameters in the script and different fields included in the isolation forest



09:30 – 09:45

Variables: How to run the program

- We can modify the following:
  - The Isolation Forest parameters

```
35  ZV_NU_ESTIMATORS=500
36  ZV_ST_MAX_SAMPLES='auto'
37  ZV_NU_CONTAMINATION=0.5
38  ZV_NU_MAX_FEATURES=1.0
39  ZV_NU_RANDOM_STATE=42
```

  - The fields that are included: Note: the fields that you include in predict must be the same as those that you put for Train.

The percentile and log of ratio are per supplier. The ABS_LOG is for the whole file.

```
31  ZV_LI_COLUMNS_NUMERIC = ['ZF_REGUH_RBETR_RANK_PCTL','ZF_REGUH_RBETR_RATIO_LOG','ZF_REGUH_RBETR_ABS_LOG']
32  ZV_LI_COLUMNS_ONE_HOT = ['REGUH_ZLAND','REGUH_ZBNKS']
33  ZV_LI_COLUMNS_HASH = ['ZF_REGUH_KUNNREMPFGLIFNR','REGUH_ZNME1','BKPF_USNAM', 'REGUH_ZBNKN']
```

09:30 – 09:45

Variables: How to run the program

Why do isolation forest when you can do ChatGPT? Or an AI to a generative model?

03

Python script: step-by-step

Vertical concatenate, statistics: mean, rank, percentile, ratio, log

Artificial intelligence: One-shot, hash, isolation forest

Data dump: pickle

09:30 – 09:45

Payment settlements data filters

- Filter on payments that are not only proposals
- Filter out key fields that don’t have values, because this can cause our statistics to not make sense

```
# 3/ Import the REGUH table
ZV_LI_NUM_COLS = [
    'REGUH_RBETR'
]
ZV_DF_REGUH = FC_IMPORT_TEXT(ZV_ST_NAME_REGUH_FILE, ZV_ST_SOURCES_FOLDER, '\t','REGUH_',ZV_LI_

# 4/ Absolute value, filter, add columns
ZV_DF_REGUH = (
    ZV_DF_REGUH
    .filter(
        (
            (PI_POLARS.col('REGUH_XVORL').str.strip_chars() != 'X') &
            (PI_POLARS.col('REGUH_ZBNKN').str.strip_chars() != '') &
            (PI_POLARS.col('REGUH_VBLNR').str.strip_chars() != '') &
            (PI_POLARS.col('REGUH_RBETR') != 0)
        )
    )
```

Statistics on numeric fields

We can decide to use statistics per supplier if we want to see if a value is very unusual compared to the rest of the values for the supplier.. Although we need to remember that very big is not very unusual, it is only if it is different to the rest… therefore, isolation forest only works well on large data sets.

```
# ---- Numeric columns ----

# Store statistics for predict
ZV_DF_REGUH_STATS = (
    ZVFCI_DF_REGUH
    .group_by('ZF_REGUH_KUNNREMPFGLIFNR')
    .agg([
        PI_POLARS.col('ZF_REGUH_RBETR_ABS').mean().alias('ZF_REGUH_RBETR_MEAN_TRAIN'),
        PI_POLARS.col('ZF_REGUH_RBETR_ABS').sort().alias('ZF_LI_REGUH_RBETR_ABS_TRAIN'),
        PI_POLARS.len().alias('ZF_REGUH_KUNNREMPFGLIFNR_COUNT_TRAIN')
    ])
)
```

Statistics on numeric fields – average and rank

Here we get the ***mean*** and the ***rank*** per supplier…

```
# Numeric statistics: Log of ratio (abs/mean) and percentile
ZV_DF_REGUH_(
    Z (function) with_columns: Any
    .with_columns(
        [
            PI_POLARS.col('ZF_REGUH_RBETR_ABS')
            .mean()
            .over('ZF_REGUH_KUNNREMPFGLIFNR')
            .alias('ZF_REGUH_RBETR_MEAN'),

            PI_POLARS.col('ZF_REGUH_RBETR_ABS')
            .rank(method='average')
            .over('ZF_REGUH_KUNNREMPFGLIFNR')
            .alias('ZF_REGUH_RBETR_RANK')
        ]
    )
```

Statistics on numeric fields – percentile and log

The ***rank*** is used to calculate the ***percentile***.

The ***mean*** is used to calculate the ***ratio***, that we can then use to calculate the ***log***.

```
            (
                PI_POLARS.col('ZF_REGUH_RBETR_RANK') /
                PI_POLARS.len()
                .over('ZF_REGUH_KUNNREMPFGLIFNR')
            )
            .alias('ZF_REGUH_RBETR_RANK_PCTL')

        ]
    )
    .with_columns(
        [
            PI_POLARS.col('ZF_REGUH_RBETR_RATIO')
            .log1p()
            .alias('ZF_REGUH_RBETR_RATIO_LOG'),

            PI_POLARS.col('ZF_REGUH_RBETR_ABS')
            .log1p()
            .alias('ZF_REGUH_RBETR_ABS_LOG')
        ]
```

Text fields– one-shot

We can use one-hot to convert text columns that do not have a lot of different values to 0s and 1s. For example, countries. There are not that many countries, so isolation forest will work better if we have one column per country with 0 or 1 in that column.

```
# --- Text columns with few different values: one-hot----
ZV_DI_ONEHOT_ALLOWED = {}
ZV_DF_REGUH_ONEHOT = ZV_DF_REGUH[ZV_LI_COLUMNS_ONE_HOT].copy()
for ZV_COL in ZV_LI_COLUMNS_ONE_HOT:
    ZV_SER = ZV_DF_REGUH_ONEHOT[ZV_COL].fillna('').astype(str)

    # Keep top 50 most frequent values (tune N)
    ZV_LI_TOP = ZV_SER.value_counts().head(50).index.tolist()

    ZV_DI_ONEHOT_ALLOWED[ZV_COL] = set(ZV_LI_TOP)

    ZV_DF_REGUH_ONEHOT[ZV_COL] = ZV_SER.apply(lambda v: v if v in ZV_DI_ONEHOT_ALLOWED[ZV

ZV_DF_REGUH_ONEHOT = PI_PANDAS.get_dummies(
    ZV_DF_REGUH_ONEHOT.copy(),
    drop_first=False
)

ZV_LI_COLUMNS_ONE_HOT_CREATED = ZV_DF_REGUH_ONEHOT.columns.to_list()
print('one-hot dataframe created')
```

Text fields - hash

We can use hash to convert text columns that have a lot of different values. For example, bank account numbers. There may be many bank account numbers, so isolation forest works better if we calculate the hash of these types of columns.

```
# --- Hash ---
# Convert each row into list of strings for hashing
ZV_DF_REGUH_HASH = ZV_DF_REGUH[ZV_LI_COLUMNS_HASH].copy()
ZV_SE_REGUH_HASH = (
    ZV_DF_REGUH_HASH
    .astype(str)
    .apply(lambda row: row.values.tolist(), axis=1)
)

ZV_OB_HASHER = PI_FEATUREHASHER(
    n_features=32,
    input_type="string"
)

ZV_SP_REGUH_HASH = ZV_OB_HASHER.transform(ZV_SE_REGUH_HASH)

ZV_DF_REGUH_HASH = PI_PANDAS.DataFrame(
    ZV_SP_REGUH_HASH.toarray(),
    index=ZV_DF_REGUH.index
)
ZV_DF_REGUH_HASH.columns = [f"ZF_HASH_{i:03d}" for i in range(ZV_DF_REGUH_HASH.shape
ZV_LI_COLUMNS_HASH_CREATED = ZV_DF_REGUH_HASH.columns.to_list()
print('hashed dataframe created')
```

Combine all tables together

If the index is correct, we can concatenate vertically (axis = 1)– we don´t even need to join.

Before doing this we also need to make sure that we did not change the order of the rows.

```
# ---Recombine----
ZV_DF_X = (
    PI_PANDAS
    .concat(
        [
            ZV_DF_REGUH_NUMERIC,
            ZV_DF_REGUH_ONEHOT,
            ZV_DF_REGUH_HASH
        ],
        axis=1
    )
    .fillna(0)
    .copy()
)
print('Reguh recombined')
```

Create and apply isolation forest object to create a model

Then we can apply the isolation forest model – the variables can be adjusted depending on our output.

These variables were created at the top of our script.

```
# Initiate isolation forest object
ZF_OB_ISF_MODEL = PI_ISOLATIONFOREST(
    n_estimators=ZV_NU_ESTIMATORS,
    max_samples=ZV_ST_MAX_SAMPLES,
    contamination=ZV_NU_CONTAMINATION,
    max_features=ZV_NU_MAX_FEATURES,
    random_state=ZV_NU_RANDOM_STATE
)

# Train isolation forest on the dataset
ZF_OB_ISF_MODEL.fit(ZV_DF_X)
print('Isolation Forest model created')
```

Save the model and information about the model - pickle

We create a dictionary of all that we have created including our model and then we pickle it.

wb means overwrite if already exists

We don´t need to put the file path if we are saving to the same folder as the programs folder.

```
# Create the artifacts dictionary
ZV_DI_ISF_ARTIFACTS = {
    'ZV_DK_ZF_REGUH_STATS': ZV_DF_REGUH_STATS,
    'ZV_DK_ZF_OB_ISF_MODEL': ZF_OB_ISF_MODEL,
    'ZV_DK_ZV_OB_HASHER': ZV_OB_HASHER,
    'ZV_DK_ZV_LI_COLUMNS_ONE_HOT_CREATED': ZV_LI_COLUMNS_ONE_HOT_CREATED,
    'ZV_DK_ZV_DI_ONEHOT_ALLOWED': ZV_DI_ONEHOT_ALLOWED,
    'ZV_DK_ZV_LI_COLUMNS_HASH_CREATED': ZV_LI_COLUMNS_HASH_CREATED,
    'ZV_DK_ZV_LI_COLUMNS_NUMERIC': ZV_LI_COLUMNS_NUMERIC,
    'ZV_DK_ZV_LI_COLUMNS_ONE_HOT': ZV_LI_COLUMNS_ONE_HOT,
    'ZV_DK_ZV_LI_COLUMNS_HASH': ZV_LI_COLUMNS_HASH,

}
print('Artifacts dictionary created')

# Save the artifacts dictionary
ZV_ST_DI_ISF_ARTIFACTS_NAME = f'ZV_DI_ISF_ARTIFACTS{ZV_ST_ISF_MODEL_NUMBER}.sav'
PI_PICKLE.dump(ZV_DI_ISF_ARTIFACTS, open(ZV_ST_DI_ISF_ARTIFACTS_NAME, 'wb'))
```

Predict using the model

Then in predict, we import the pickle, get the model and use the model to score our new data set.

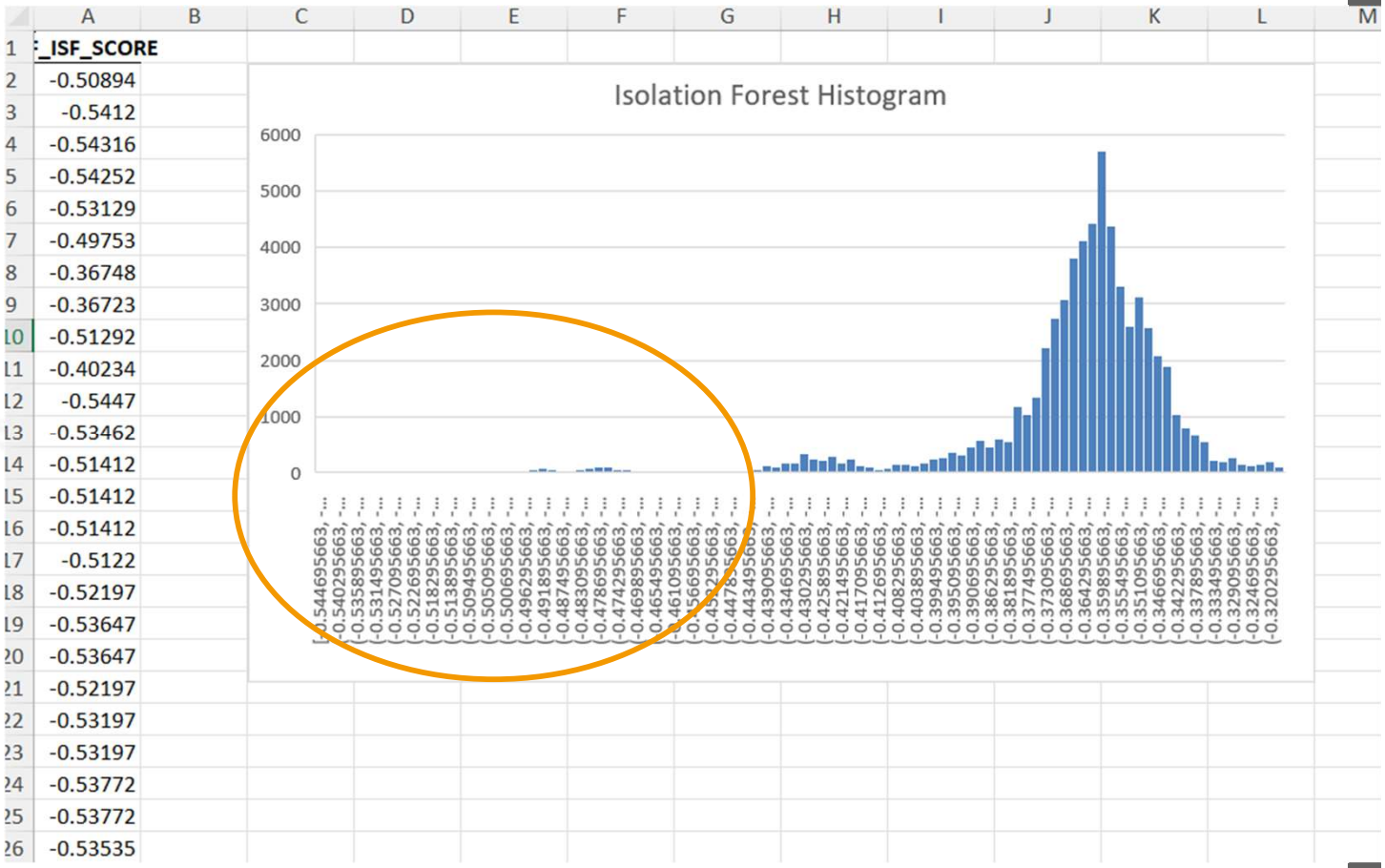
```
# --- Score using X_NEW ---
ZV_DF_REGUH_ISF = ZV_DF_REGUH.copy()
ZV_DF_REGUH_ISF["ZF_ISF_ANOMALY"] = ZF_OB_ISF_MODEL.predict(ZV_DF_X_NEW)
ZV_DF_REGUH_ISF["ZF_ISF_SCORE"]   = ZF_OB_ISF_MODEL.score_samples(ZV_DF_X_NEW)

# Export result to csv file
FC_EXPORT_EXCEL_PANDAS(ZV_DF_REGUH_ISF, ZV_04_C_PHASE_FOLDER, ZV_ST_NAME_RESULTS_FILE)
```

We can use Excel to plot a histogram also

In Excel, a simple histogram shows us how unusual the transactions are.

Copy the Isolation Forest score to a new sheet
Highlight the column
Click on Insert->
Statistics chart ->
histogram

The ***rare*** transactions are the ones that have a more negative score



	A
1	_ISF_SCORE
2	-0.50894
3	-0.5412
4	-0.54316
5	-0.54252
6	-0.53129
7	-0.49753
8	-0.36748
9	-0.36723
10	-0.51292
11	-0.40234
12	-0.5447
13	-0.53462
14	-0.51412
15	-0.51412
16	-0.51412
17	-0.5122
18	-0.52197
19	-0.53647
20	-0.53647
21	-0.52197
22	-0.53197
23	-0.53197
24	-0.53772
25	-0.53772
26	-0.53535

09:30 – 09:45


Questions?