Bank of England data scientist role Assignment

Giorgio Miraglia

Table of contents

Introduction and report overview

This report identifies outlier insurance firms based on three main criteria: firm size, whether the business profile of the firm is changing, and being an outlier from the 'norm'. The criteria are used separately from one another, each forming a section of this report. In each section the report identifies which firms are most likely to be outlier firms and therefore require increased supervisory resources from the PRA. There are what seem to be significant data quality issues, which are addressed after this introduction.

Whilst in sections one and two, identifying outlier firms by size and changing business profile, I utilise primarily univariate approaches, in section three I identify 'outliers from the norm' by fitting isolation forests on the whole feature set. This yields a quick, comprehensive, and explainable method for detecting outliers non parametrically. Finally in the annex, I backup my findings in section three by detecting the same outliers using density-based clustering and dimensionality reduction techniques.

Task 1

Data preperation and inspection

Config and packages

```
import pandas as pd
import re
import pickle
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from IPython.display import display, Markdown
pd.options.mode.chained_assignment = None  # default='warn'

#isolation forest
from sklearn.ensemble import IsolationForest

#dbscan
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

#shapley values
import shap

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# Load the Excel file
file_path = 'data/DataScientist_009749_Dataset.xlsx'
```

Read in, merge and clean data

```
def process_sheet(path, sheet, type_name):
    '''
    Function that reads in sheet, cleans it and pivots the data long,
    preparing data for merge with other sheets.
```

```python
    '''
    df = pd.read_excel(path, sheet_name=sheet)

    #extract years
    yearlist = list(df.iloc[0,1:].str.replace('YE', ''))

    #remove first row
    df = df.iloc[1:]

    #pivot long and keep year information using regex
    pattern = r'([^.]*)'
    cleancols = [re.search(pattern, input).group(1) for input in list(df.columns)][1:]
    newcols = [f'{c}-{y}' for c, y in zip(cleancols, yearlist)]
    df.columns = ['Firm'] + newcols

    #melt
    df = pd.melt(df, id_vars=['Firm'],
            var_name='metric', value_name='value')
    df[['metric', 'year']] = df['metric'].str.split('-', expand=True)
    df['type'] = type_name
    df = df[['Firm', 'year', 'type', 'metric', 'value']]

    return df

df1 = process_sheet(path = file_path, sheet = 0, type_name = 'general')
df2 = process_sheet(path = file_path, sheet = 1, type_name = 'underwriting')

#clean and concat dataset
df = pd.concat([df1, df2])

#drop firm for which all values are 0
sumdf = df.groupby('Firm', as_index = False)['value'].sum()
drop_firms = list(sumdf[sumdf.value == 0]['Firm'].unique())
df = df[~df['Firm'].isin(drop_firms)]

#set columnall types
df['year'] = df['year'].astype(np.int64)
df['value'] = df['value'].astype('float64')
df['Firm'] = df['Firm'].astype('str')
df['metric'] = df['metric'].astype('str')
df['type'] = df['type'].astype('str')
```

```
#save
with open('mergeddf.pickle', 'wb') as handle:
    pickle.dump(df, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

Check for missing values

As a basic data check, we inspect missing values from the dataframe. Most of these, seen below, pertain to firms which only have data in one sheet. For consistency in analysis, I drop these observations.

```
# Pivot the DataFrame to long format
dfwide = pd.pivot_table(df, values='value', index=['Firm', 'year'], columns='metric').rese

# Rename the columns for clarity
dfwide.columns.name = None

#drop firms that only appear in one sheet
dfwide = dfwide.dropna()
```

Summary stats and erroneous outliers

We inspect the distributions of the variables. (Due to formatting issues, this can be found in the original quarto file, but not in this pdf). Looking in particular at the min and max values, we immediately notice what appear to be unfeasible numbers, which raise questions on the quality of the data being used. For example, negative GWP numbers and enormous ratios. These erroneous outliers will skew the averages and standard deviations - this is important because my method for detecting outliers in the first two sections depends on these.

Using the variables in the assignment description document, I create general rules to filter out possible instances of misreporting. These include:

- 'Ratio' variables with earned premiums in the denominator cannot take values greater than ten and must take values greater than zero.
- GWP and NWP must be greater than 0.

As shown below, this results in 12% of observations being removed from the data. This is a high number and warrants further investigation. If one were more strict in identifying these types of observations we would remove close to half the data.

```
##could change or delete this, then modify underlying so it affects all downstream code ch
mask = ((dfwide['Gross claims incurred (£m)'] < 0) | #insurance firms should be incurring
```

```
                (dfwide['Net combined ratio'] < 0) | (dfwide['Net combined ratio'] > 10) | #
                (dfwide['SCR coverage ratio'] < 0) | #ratio should be greater than zero, nei
                (dfwide['NWP (£m) '] < 0) |
                (dfwide['GWP (£m)'] < 0))

missreported = dfwide[mask].shape[0]

print(f'{round(100 * missreported / dfwide.shape[0], 2)}% of observations identified as mi

dfwide = dfwide[~mask]
```

```
12.01% of observations identified as misreporting and removed.
```

There is scope to remove misreported figures more surgically by using domain knowledge, understanding the reporting requirements on firms, and looking for further patterns in the suspicious observations.

```
dfwide.describe()
```

	year	EoF for SCR (£m)	Excess of assets over liabilities (£m) [= equity]	GWP (£m)	C
count	1421.000000	1421.000000	1421.000000	1421.000000	1
mean	2018.039409	478.927971	535.677040	973.372234	3
std	1.412667	2073.514047	2175.169453	4931.338015	1
min	2016.000000	0.000000	0.000000	0.000000	-
25%	2017.000000	2.829564	3.720507	0.000000	0
50%	2018.000000	24.660971	29.575202	11.471268	3
75%	2019.000000	162.428978	167.731622	217.342744	1
max	2020.000000	36644.404797	26705.042053	74078.635849	1

Firm Size outliers

In this section we focus on finding firms that are outliers in size. For finding these firms, we hone in on four metrics which together capture the size of a firm. These are: Gross written premium, net written premium, total assets, and total liabilities. We focus on the latest data, 2020. Furthermore, for all of these variable we consider their log distributions rather than the nominal values, as earnings/revenues data, like incomes, are likely nonlinear.

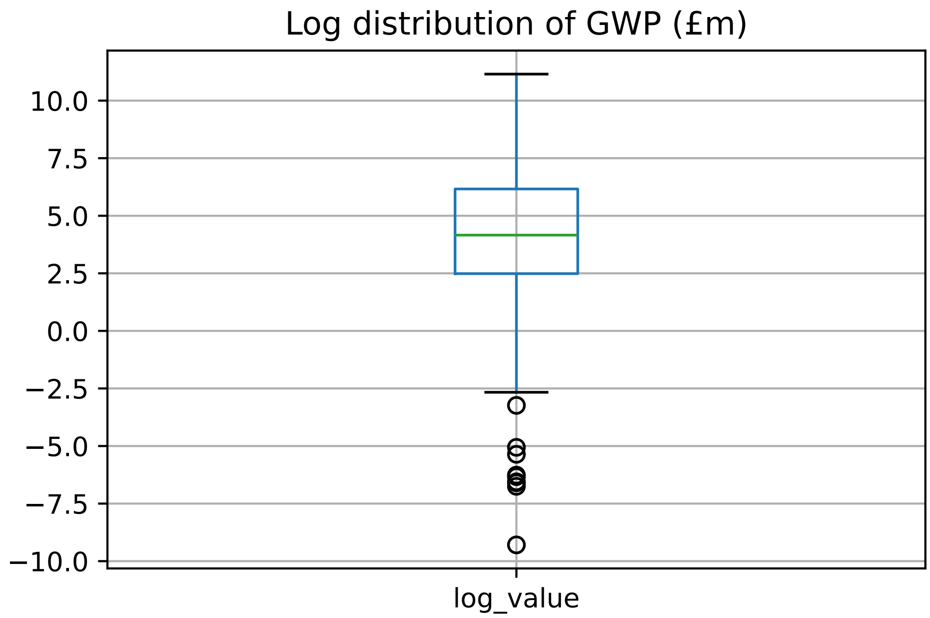
```
def outliers_and_plot(df, metric, standard_deviations):
```

```
    dfsize = df[df[metric] > 0]
    dfsize['log_value'] = np.log(dfsize[metric])
    #dfsize['log_value'].hist(bins = 50)
    #fig = dfsize['log_value'].plot.density(figsize = (8,5))
    fig = dfsize[['log_value']].boxplot()
    plt.title(f'Log distribution of {metric}')

    ##do this for the four metrics first

    ##then identify outliers as per below method

    mean = dfsize['log_value'].mean()
    sd = dfsize['log_value'].std()

    #gives top 5% of firms according to log normal distribution
    firms = dfsize[dfsize['log_value'] > mean + standard_deviations*sd][['Firm', metric]]
    firms.rename(columns = {'Firm' : f'{metric} outliers'}, inplace = True)

    return fig, firms
```

In the below tabs, I show the boxplot distributions of the logs of these variables. And below these are the firms identified as outliers for that variable, defined as being 1.64 times the standard deviation above the mean. Assuming a normal distribution, this is identifying roughly the top 5% of firms.

GWP

```
fig, gwpoutliers = outliers_and_plot(dfwide[(dfwide['year'] == 2020)], 'GWP (£m)', 1.64)
```



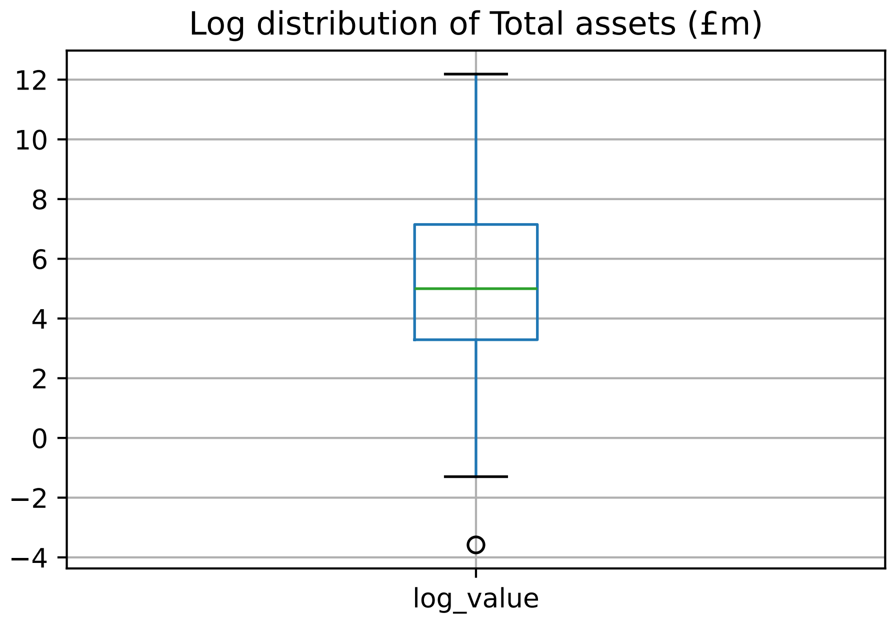

```
print(f'{gwpoutliers.shape[0]} outliers found:')

display(
    Markdown(
        gwpoutliers.to_markdown()
    )
)
```

```
4 outliers found:
```

	GWP (£m) outliers	GWP (£m)
614	Firm 210	69697.9
1174	Firm 311	24251.5
1304	Firm 34	19275
1939	Firm 7	16183.6

Total assets (£m)

```
fig, assetsoutliers = outliers_and_plot(dfwide[(dfwide['year'] == 2020)], 'Total assets (£
```




```
print(f'{assetsoutliers.shape[0]} outliers found:')

display(
    Markdown(
        assetsoutliers.to_markdown()
    )
)
```

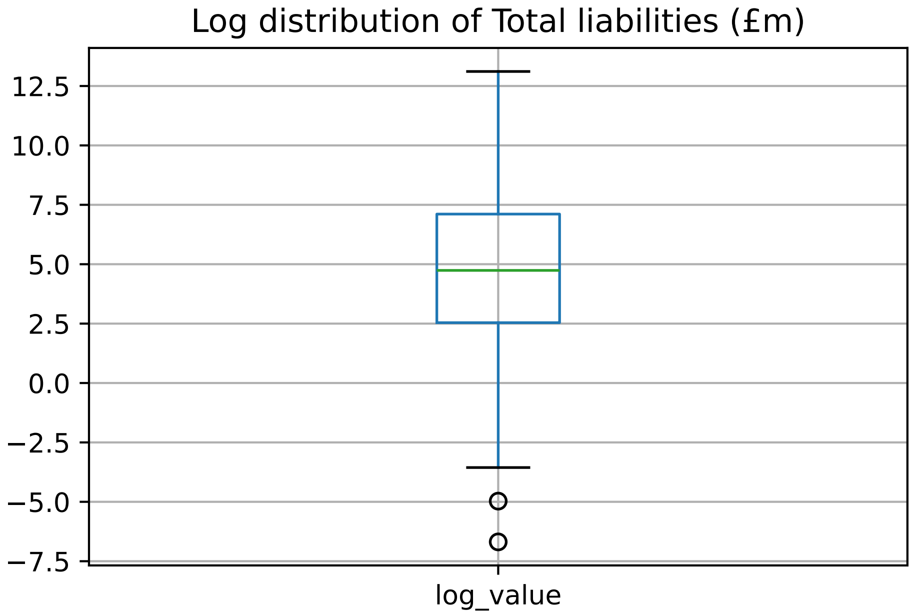
```
16 outliers found:
```

	Total assets (£m) outliers	Total assets (£m)
9	Firm 10	195836
39	Firm 105	105213
544	Firm 199	155762
599	Firm 208	54768.7
614	Firm 210	160519
814	Firm 247	100085
884	Firm 26	83195.5
974	Firm 276	56366.9
1094	Firm 298	68871.8
1109	Firm 30	44306.9

	Total assets (£m) outliers	Total assets (£m)
1174	Firm 311	61836.6
1304	Firm 34	160124
1839	Firm 51	32271.7
1884	Firm 6	107548
1939	Firm 7	58820.1
1959	Firm 73	107523

Total liabilities (£m)

```
fig, liabilitiesoutliers = outliers_and_plot(dfwide[(dfwide['year'] == 2020)], 'Total liab
```



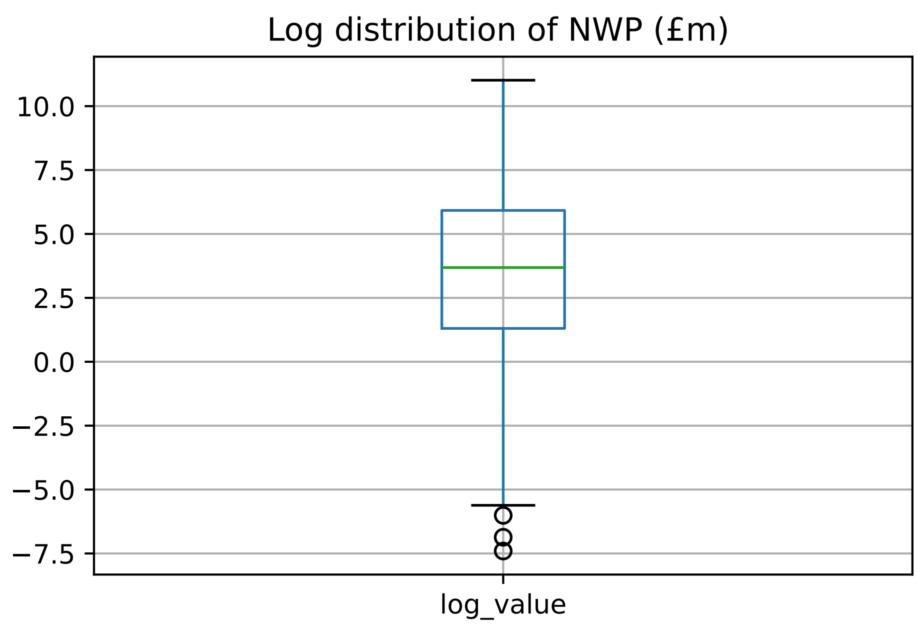

```
print(f'{liabilitiesoutliers.shape[0]} outliers found:')

display(
    Markdown(
        liabilitiesoutliers.to_markdown()
    )
)
```

```
11 outliers found:
```

	Total liabilities (£m) outliers	Total liabilities (£m)
9	Firm 10	254270
39	Firm 105	175128
599	Firm 208	60409.2
814	Firm 247	79654.9
974	Firm 276	49396.1
1094	Firm 298	118465
1109	Firm 30	99549.6
1174	Firm 311	494499
1304	Firm 34	182264
1939	Firm 7	178293
1959	Firm 73	55884

NWP (£m)

```
fig, nwpoutliers = outliers_and_plot(dfwide[(dfwide['year'] == 2020)], 'NWP (£m) ', 1.64)
```




```
print(f'{nwpoutliers.shape[0]} outliers found:')

display(
```

```
    Markdown(
        nwpoutliers.to_markdown()
    )
)
```

```
5 outliers found:
```

	NWP (£m) outliers	NWP (£m)
544	Firm 199	13133.5
614	Firm 210	60700
829	Firm 25	9765.61
884	Firm 26	16395.7
1174	Firm 311	14566.3

One firm, 311 , appears as an outlier across all four metrics. Supervisory resources should likely be dedicated to this firm. It is also worth noting that firm 34 appears in three out of four categories. However, considering the poor quality of the data, this may also be an instance of misreporting.

```
alloutliers = pd.concat([nwpoutliers, liabilitiesoutliers, assetsoutliers, gwpoutliers], a
display(
    Markdown(
        alloutliers.dropna()
.to_markdown()
    )
)
```

	NWP (£m) outliers	NWP (£m)	Total liabilities (£m) outliers	Total liabilities (£m)	Total assets (£m) outliers	Total assets (£m)	GWP (£m) outliers	GWP (£m)
1174	Firm 311	14566.3	Firm 311	494499	Firm 311	61836.6	Firm 311	24251.5

It is also of interest that not all of the largest firms by GWP are the largest by NWP. This suggests some of these firms will have very significant reinsurance activity. This may warrant further investigation from a business model perspective. It would also be interesting here to investigate whether the largest firms are also the ones incurring the most claims in nominal terms. If the firms incurring the most claims are not also the largest, perhaps those firms are at risk of failing and require supervisory attention.

Changing business profile

Like in section one, to identify outliers in changing business profile I focus on four variables. I think these have the most potential for harm. They are:

- Change in SRC coverage ratio: which in the case of rapid deterioration warrants the PRA's attention.
- % change in equity: if there are large decreases in equity this can indicate problems in the firm's balance sheet and potential future insolvency (in severe cases).
- Change in net combined ratio: A large increase in net combined ratio means decreasing profitability which can be a prudential risk.
- Change in pure gross claims ratio: similar to changes in net combined ratio, if there is a sharp increase in claims as a ratio of earned premiums that can be a significant source of cost pressure on the insurance firm.

Furthermore, I decide to focus only on the above variables for the 2020 period, which is the most recent in the data. Outliers in changing business profiles in 2020 should be more important than outliers from previous years for supervisors.

Data prep and outlier function:

```
calc_cols = ['EoF for SCR (£m)',
                        'Excess of assets over liabilities (£m) [= equity]',
                        'GWP (£m)',
                        'Gross BEL (inc',
                        'Gross claims incurred (£m)',
                        'Gross combined ratio',
                        'Gross expense ratio',
                        'NWP (£m) ',
                        'Net BEL (inc',
                        'Net combined ratio',
                        'Net expense ratio',
                        'Pure gross claims ratio',
                        'Pure net claims ratio',
                        'SCR (£m)',
                        'SCR coverage ratio',
                        'Total assets (£m)',
                        'Total liabilities (£m)']

dfchange = dfwide.copy()

firmcol, yearcol = dfchange['Firm'], dfchange['year']
```

```
#diff dataframe
dfdiff = dfchange.drop('year', axis = 1).groupby(['Firm'], as_index = False).diff()
dfdiff['Firm'] = firmcol
dfdiff['year'] = yearcol
dfdiff = dfdiff[['Firm', 'year'] + calc_cols]
dfdiff = dfdiff.dropna()


#percentage diff dataframe
dfpercdiff = dfchange.drop('year', axis = 1).groupby(['Firm'], as_index = False).pct_chang
dfpercdiff['Firm'] = firmcol
dfpercdiff['year'] = yearcol
dfpercdiff = dfpercdiff[['Firm', 'year'] + calc_cols]

dfpercdiff = dfpercdiff.dropna()
dfpercdiff.replace([np.inf, -np.inf], np.nan, inplace=True) # remove infinite values which


#lets hone in on changes over over that past year, this is important to supervisors
dfchange2020 = dfdiff[(dfdiff['year'] == 2020)]

dfchangepercent2020 = dfpercdiff[(dfpercdiff['year'] == 2020)]

def sd_outliers(df, var, standard_deviations, greater_than_or_less_than):
    '''
    Find outlier firms, defined as, depending on the value of 'greater_than_or_less_than'
    as firms with a var value greater than or less than three times the standard deviation
    above or below the mean.
    '''

    if greater_than_or_less_than == '>':
        a = df[(df[var] > (df[var].mean() + standard_deviations * df[var].std()))][['Firm'

        a['threshold percentile'] = stats.percentileofscore(df[var],
                                                            (df[var].mean() + standard_deviati

    else:
        a = df[(df[var] < (df[var].mean() - standard_deviations * df[var].std()))][['Firm'

        a['threshold percentile'] = stats.percentileofscore(df[var],
                                                            (df[var].mean() - standard_deviati
```

```
    a['threshold'] = (df[var].mean() + standard_deviations * df[var].std())

    a = a.rename(columns = {'firm' : var + '_outliers'})

    return a
```

In the below tabs, I show the density plots for the variables of interest. As also shown by the summary table below, all four variables are centred around zero (50th percentile), which means on average firms did not change. However, the mean changes in the combined ratio and pure gross claims ratios are negative, while positive for SCR coverage ratio and equity. These should be considered positive moves in all four variables. This feels inconsistent with the wider macroeconomic environment in 2020 and likely high number and unexpected pandemic insurance claims. This warrants further investigation.

Another point of interest is how we treat the percentage change in equity variable. The minimum value is -100%, (percentage table not shown here) which is a significant deterioration of the balance sheet if true. Given this bounded nature of the variable, I choose -100% to be the outlier threshold for the % change in equity variable. The other thresholds are:

- SRC coverage ratio: less than 1.64 x std below the mean
- Net combined ratio: more than 1.64 x std above the mean
- Pure gross claims ratio: more than 1.64 x std above the mean

The thresholds are in the directions that are potentially harmful. It seems likely that supervisors would not necessarily be concerned if an insurance firm has a sharp increase in their SRC coverage ratio.

```
display(
    Markdown(
        dfchange2020[['SCR coverage ratio', 'Net combined ratio',
        'Pure gross claims ratio', 'Excess of assets over liabilities (£m) [= equity]']].d
    )
)
```

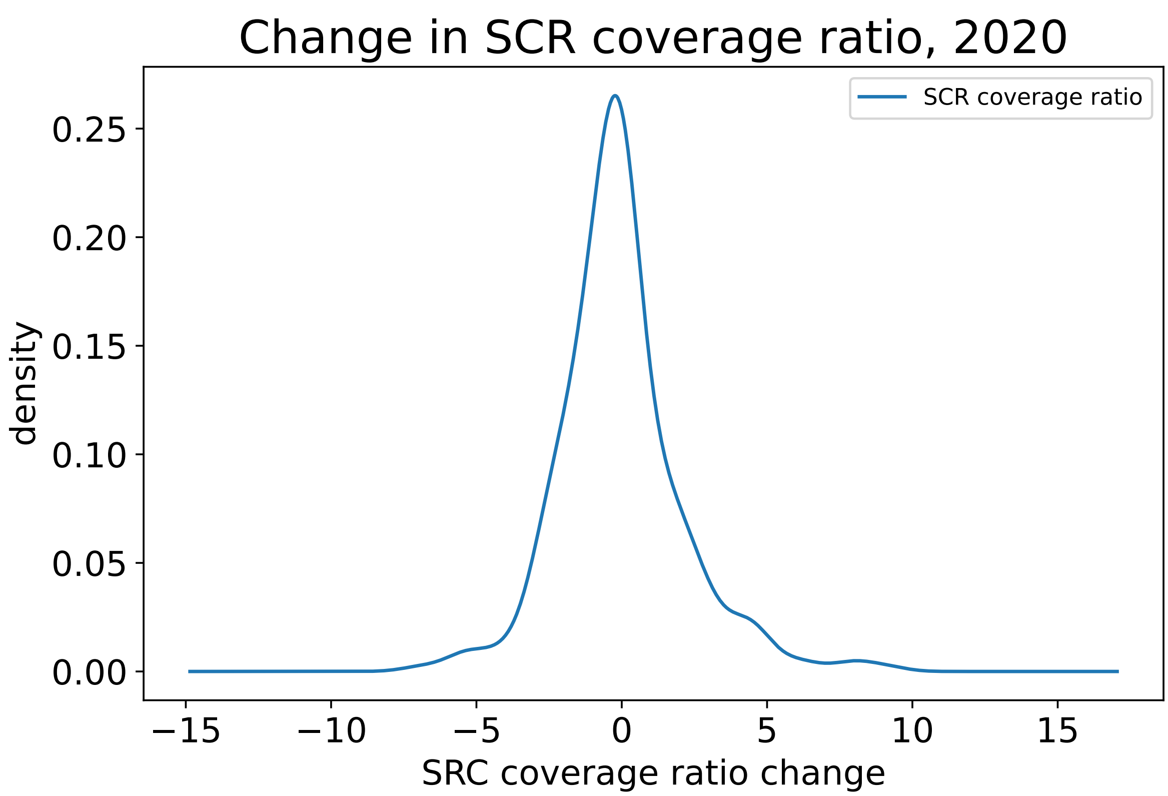
	SCR coverage ratio	Net combined ratio	Pure gross claims ratio	Excess of assets over liabilities (£m) [= equity]
count	293	293	293	293
mean	15.79	0.0463258	20.5125	-46.8112
std	291.82	0.907844	300.678	1531.82
min	-125.187	-3.42867	-2.75507	-11279.3
25%	-1.25853	0	-0.0623387	-22.3769
50%	0	0	0	0
75%	0.835918	0.155572	0.112608	9.81019

	SCR coverage ratio	Net combined ratio	Pure gross claims ratio	Excess of assets over liabilities (£m) [= equity]
max	4990.91	7.56652	5061.22	8370.77

SRC coverage ratio

```
dfchange2020[(dfchange2020['SCR coverage ratio'] > -10) & (dfchange2020['SCR coverage rati
plt.title('Change in SCR coverage ratio, 2020', size = 20)
plt.xlabel('SRC coverage ratio change', size = 15)
plt.ylabel('density', size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.suptitle('')
```

```
Text(0.5, 0.98, '')
```



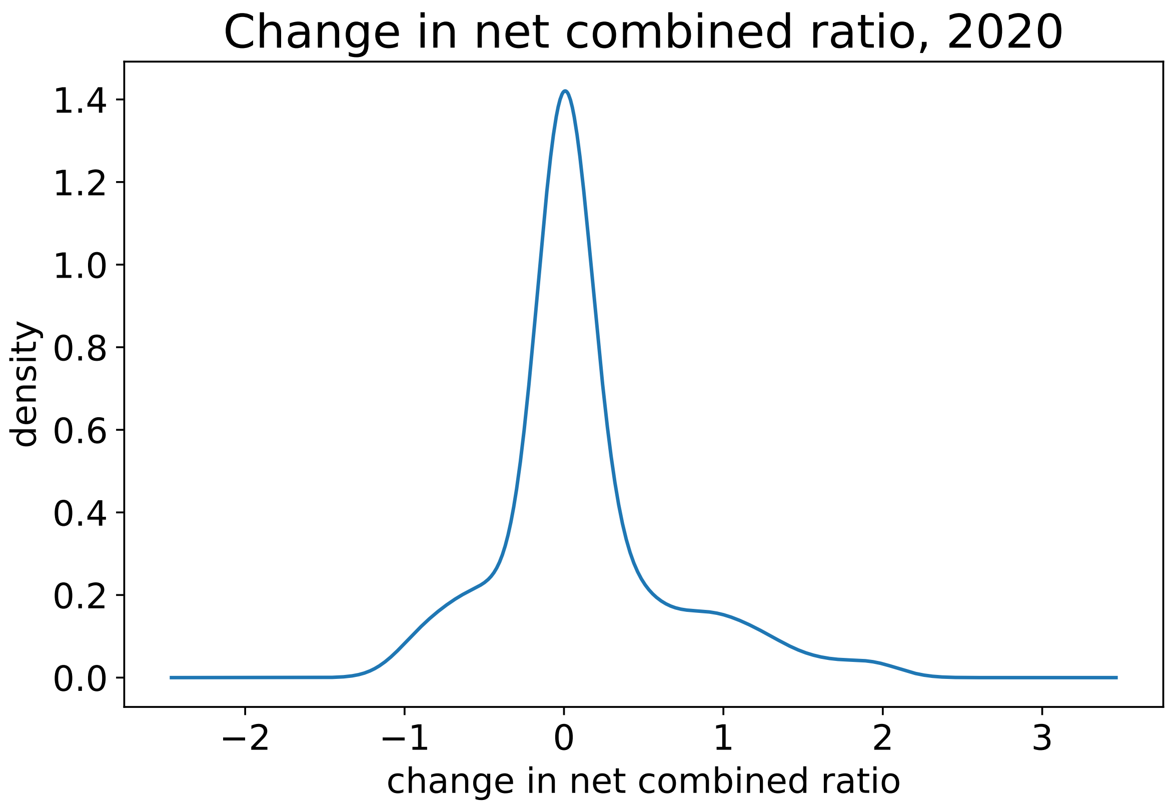
```python
src_outliers = sd_outliers(dfchange2020[(dfchange2020['SCR coverage ratio'] < 1000) &
                (dfchange2020['SCR coverage ratio'] > -40)], 'SCR coverage ratio', 1.64, '<')

display(
    Markdown(
        src_outliers.to_markdown()
    )
)
```

	Firm	SCR coverage ratio	threshold percentile	threshold
144	Firm 125	-12.9557	3.11419	7.71005
354	Firm 163	-23.5072	3.11419	7.71005
379	Firm 168	-12.1273	3.11419	7.71005
439	Firm 18	-26.1733	3.11419	7.71005
474	Firm 186	-32.1671	3.11419	7.71005
499	Firm 190	-11.3246	3.11419	7.71005
1214	Firm 319	-32.7085	3.11419	7.71005
1229	Firm 321	-21.2858	3.11419	7.71005
1864	Firm 56	-20.435	3.11419	7.71005

Net combined ratio

```python
dfchange2020[(dfchange2020['Net combined ratio'] >= -1) &
          (dfchange2020['Net combined ratio'] < 2)]['Net combined ratio'].plot.density(fig
plt.title('Change in net combined ratio, 2020', size = 20)
plt.xlabel('change in net combined ratio', size = 15)
plt.ylabel('density', size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.suptitle('')
```

```
Text(0.5, 0.98, '')
```




```
netcomb_outliers = sd_outliers(dfchange2020[(dfchange2020['Net combined ratio'] > -10000)
    (dfchange2020['Net combined ratio'] < 50)],'Net combined ratio', 1.64, '>')

display(
    Markdown(
        netcomb_outliers.to_markdown()
    )
)
```

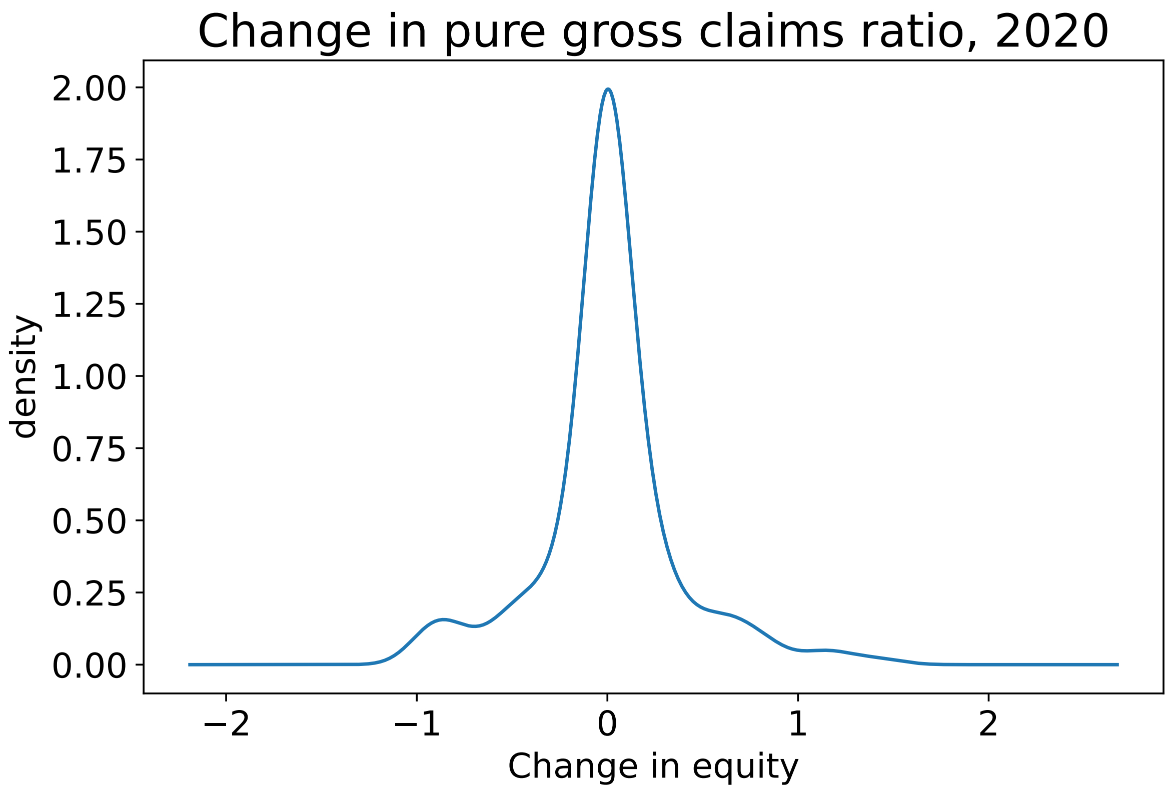
	Firm	Net combined ratio	threshold percentile	threshold
99	Firm 116	1.58919	95.9044	1.53519
249	Firm 144	2.45377	95.9044	1.53519
259	Firm 146	5.12573	95.9044	1.53519
489	Firm 189	1.59547	95.9044	1.53519
574	Firm 203	2.41665	95.9044	1.53519
704	Firm 227	2.39621	95.9044	1.53519
1059	Firm 291	1.86557	95.9044	1.53519

	Firm	Net combined ratio	threshold percentile	threshold
1139	Firm 305	1.96533	95.9044	1.53519
1509	Firm 39	7.56652	95.9044	1.53519
1564	Firm 40	1.89416	95.9044	1.53519
1844	Firm 52	1.98173	95.9044	1.53519
2064	Firm 92	1.64216	95.9044	1.53519

Pure gross claims ratio

```
dfchange2020[(dfchange2020['Pure gross claims ratio'] >= -1) &
          (dfchange2020['Pure gross claims ratio'] < 2)]['Pure gross claims ratio'].plot.d
plt.title('Change in pure gross claims ratio, 2020', size = 20)
plt.xlabel('Change in equity', size = 15)
plt.ylabel('density', size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.suptitle('')
```

```
Text(0.5, 0.98, '')
```




```
pure_claims_outliers = sd_outliers(dfchange2020[(dfchange2020['Pure gross claims ratio'] >
            (dfchange2020['Pure gross claims ratio'] < 4)], 'Pure gross claims ratio', 1.

display(
    Markdown(
        pure_claims_outliers.to_markdown()
    )
)
```

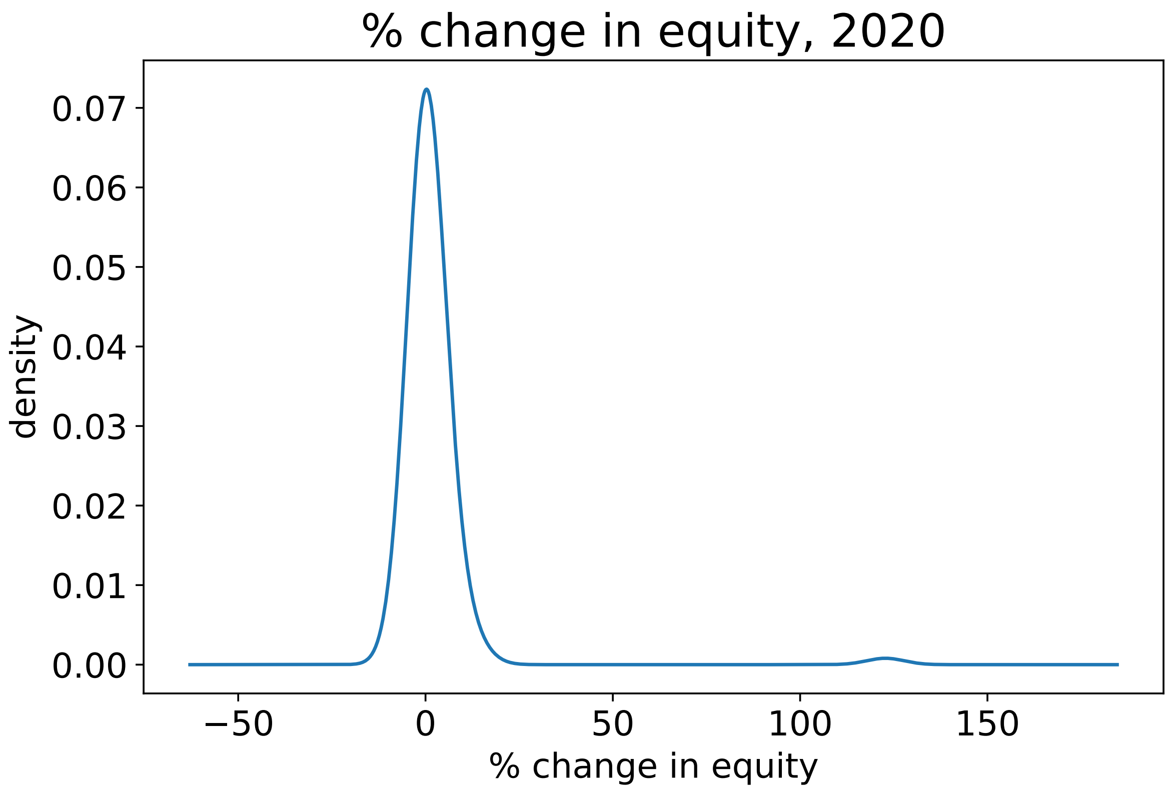
	Firm	Pure gross claims ratio	threshold percentile	threshold
209	Firm 137	1.12932	95.2206	0.744007
259	Firm 146	1.1927	95.2206	0.744007
584	Firm 205	1.45815	95.2206	0.744007
634	Firm 214	3.7451	95.2206	0.744007
639	Firm 215	1.33464	95.2206	0.744007
849	Firm 253	0.825213	95.2206	0.744007
894	Firm 261	1.06068	95.2206	0.744007

	Firm	Pure gross claims ratio	threshold percentile	threshold
954	Firm 272	0.802484	95.2206	0.744007
999	Firm 280	0.846725	95.2206	0.744007
1034	Firm 287	1.16712	95.2206	0.744007
1139	Firm 305	0.79001	95.2206	0.744007
1214	Firm 319	2.1149	95.2206	0.744007
1844	Firm 52	0.883558	95.2206	0.744007

Equity

```
#exclude problem observations
dfchangepercent2020[(~dfchangepercent2020['Excess of assets over liabilities (£m) [= equit
          (dfchangepercent2020['Excess of assets over liabilities (£m) [= equity]'] >= -10
          (dfchangepercent2020['Excess of assets over liabilities (£m) [= equity]'] < 400)
plt.title('% change in equity, 2020', size = 20)
plt.xlabel('% change in equity', size = 15)
plt.ylabel('density', size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.suptitle('')
```

```
Text(0.5, 0.98, '')
```




```
#equity outliers
equityoutliers = dfchangepercent2020[dfchangepercent2020['Excess of assets over liabilitie

display(
    Markdown(
        equityoutliers.to_markdown()
    )
)
```

	Firm
354	Firm 163
389	Firm 17
399	Firm 171
514	Firm 193
789	Firm 242
794	Firm 243
954	Firm 272
1054	Firm 290

	Firm
1144	Firm 306
1209	Firm 318
1739	Firm 44

Interestingly, although a large number of firms have been identified as outliers across the metrics, there is zero overlap of firms identified in each of these. Changes in these metrics may be capturing very different types of changes in a business, and so it seems as though no firms are experiencing great changes across more than one of these.

Outliers from the 'norm'

In this section I identify outliers using a non-parametric, multivariate approach: isolation forests. This is an anomaly detection algorithm that works by randomly selecting features and partitioning the data points into trees. The algorithm measures the number of partitions required to isolate a data point, and anomalies are identified as those points that require fewer partitions, indicating their uniqueness and potential to be outliers.

After fitting the isolation forest on the firm data, it provides a 'score' which allows me to rank the outliers. I specify the contamination parameter as being 0.05, which means we expect 5% of observations to be outliers. I chose this amount because I think it is reasonable that supervisors would want to target a small but not insignificant number of firms.

I then use Shapley values to explain what is driving the 'outlierness' for the firms we find via this method. This is a technique used to quantify the contribution of individual features to a specific prediction.

```
df_forest = dfwide[dfwide['year'] == 2020]

# Initialize the Isolation Forest model
iso_forest = IsolationForest(contamination = 0.01,
                             random_state=42,
                             bootstrap = True)  # Adjust contamination as needed

iso_forest.fit(df_forest[calc_cols])

df_forest['outlier'] = iso_forest.predict(df_forest[calc_cols])

#score data
df_forest['outlier_score'] = iso_forest.score_samples(df_forest[calc_cols])
```

```
df_forest.sort_values('outlier_score')[['Firm', 'outlier_score']].head()
```

	Firm	outlier_score
39	Firm 105	-0.763590
614	Firm 210	-0.710597
1174	Firm 311	-0.692270
1304	Firm 34	-0.682476
2099	Firm 99	-0.633736

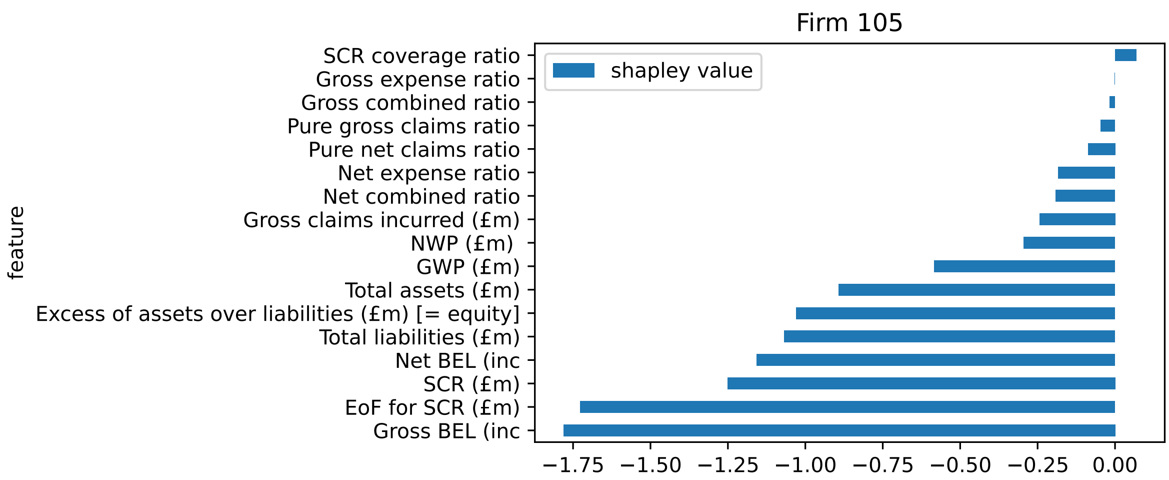
A score closer to -1 indicates an observation being more of an outlier.

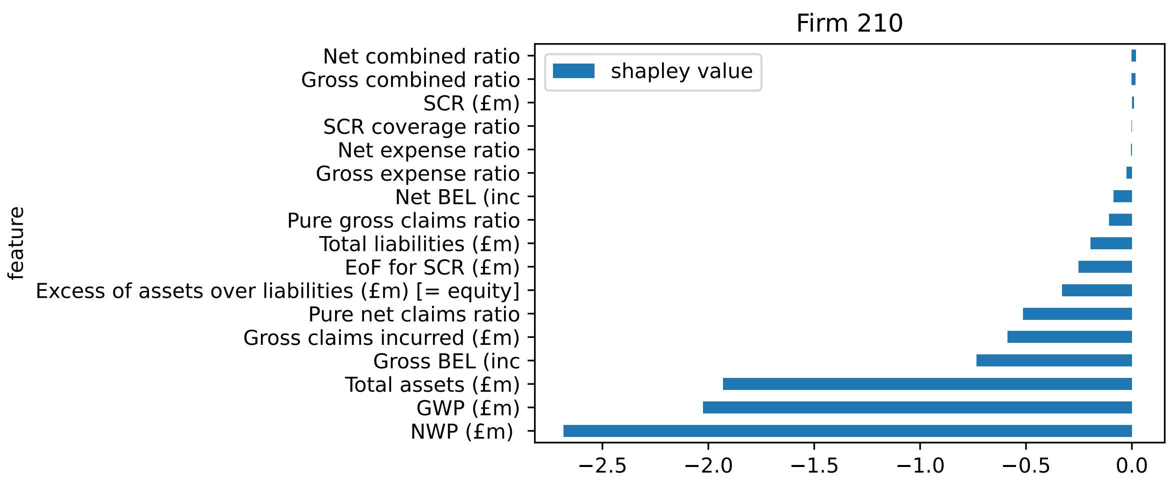
Below, we see the contributions to the individual outlier scores for the top three firms. A negative value for a feature indicates that including it in the model is driving the outlier score down for that firm, towards the -1 threshold. This most likely means it is an outlier within that feature. And inversely, a positive score indicates that the firm is not an outlier for that feature.

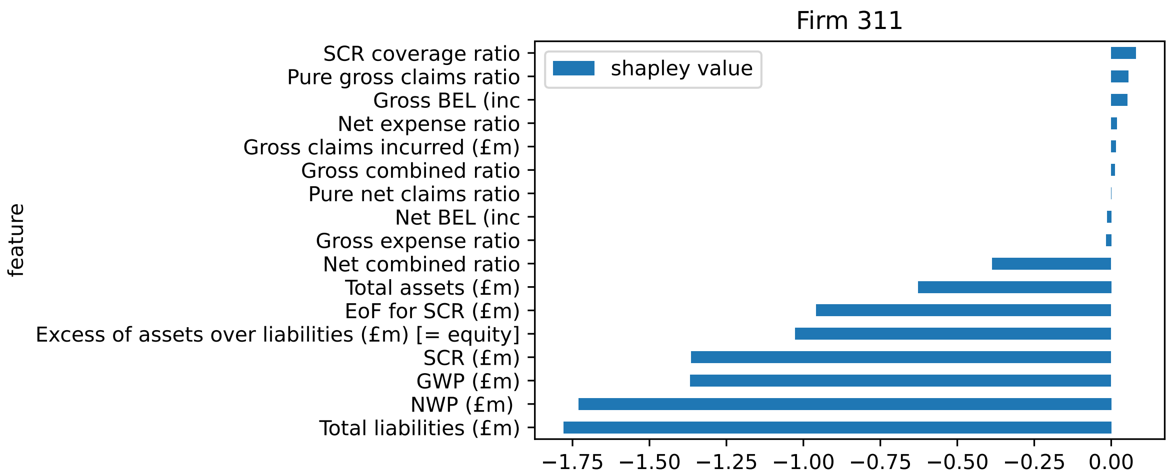
As can be seen for firm 105, most features contribute negatively towards the outlier score, most of all the Gross BEL and EoF for SCR features. There is a chance that the top one or two firms identified as outliers via this method could simply be instances of misreporting. Supervisors should be wary of this. Looking at the shapley values for the top three outlier firms, we notice that although there is some variance in which features are driving 'outlierness' across the firms, measures of size, including GWP and NWP feature in all three. Perhaps if we consider the log distributions of these variables here, as in section 1, these results will change.

```
explainer = shap.Explainer(iso_forest, df_forest[calc_cols])

#get outlier firms, could use the score instead
outlier_df = df_forest[df_forest['outlier'] == -1]

firm_list = list(outlier_df['Firm'])

outliershaps = dict()

for i, firm in enumerate(firm_list):

    shap_values = explainer.shap_values(outlier_df[calc_cols].iloc[i])

    outliershaps[firm_list[i]] = {'feature' : calc_cols,
                                  'shapley value' : list(shap_values)}
```

```
for firm in outliershaps.keys():

    pd.DataFrame.from_dict(outliershaps[firm]).sort_values('shapley value').plot(kind = 'b
                                                        x = 'feature',
                                                        y = 'shapley value',
                                                        title = f'{firm}')
```







Task 2

Annex

In this section, I locate outliers in the data using visual inspection of a two-dimensional representation of the data, and by using density-based clustering (DBSCAN). I'm interested in seeing whether these methods will produce similar outliers to those found in the 'outlier from norm' section of the report, or if they can be used to glean any additional insights from the data. Although they work quite differently, all three have largely overlapping results. This is reassuring and gives robustness to my findings.

Dimensionality reduction

Below you can see a two dimensional representation of the data using PCAs. The first thing of note here is that the firs two principal components account for less than half of the total variance in the data. This is surprisingly low and suggests the relationship between the various features is somewhat orthogonal. (Given the random multiplier explanation in the assignment this does in fact make sense). We would expect, in the real world, that many of these feature should be correlated strongly to one another. For example NWP should be proportional to GWP. Similarly, GWP is a denominator for some of the ratios, so should have a clear relationship with those as well. With real world data, the high orthogonality is suggestive of data quality issues.

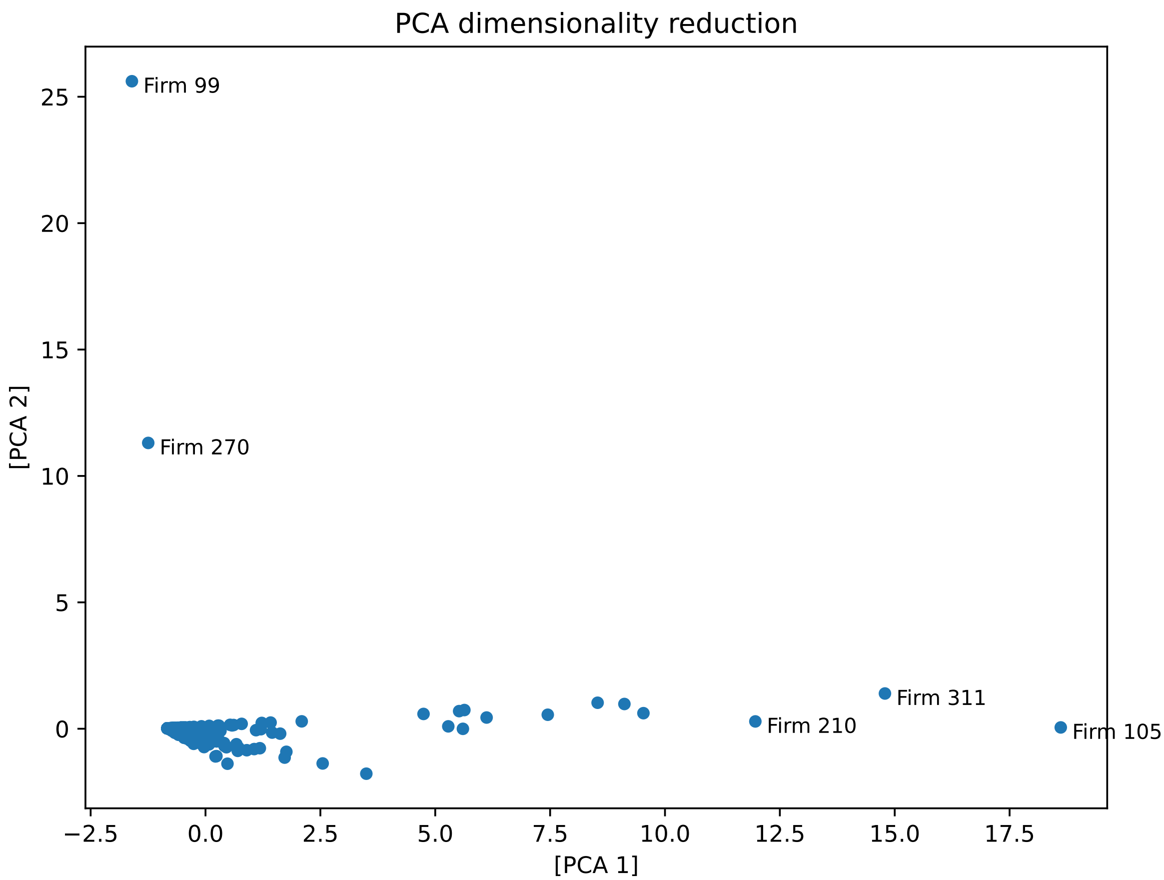
```
#Using PCA decompositions
train = dfwide[(dfwide['year'] == 2020)][calc_cols]
train = StandardScaler().fit_transform(train) #normalise data
```

```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca.fit(train)
print(f'Total explained variance of first two principal components is {100 * sum(pca.expla
##low explained variance ratio, indicates relative uncorrelatedeness of metrics, which is
```

```
Total explained variance of first two principal components is 43.26032566337855%
```

The two-dimensional representation below can be a useful way to visually locate outliers. I've labelled these in the chart. Although the PCAs will not have a direct interpretation for supervisors, I would suggest supervisors inspect the labelled firms in some more depth, and investigate what is happening with them.

```python
#plotting reduced dimensions
reduced_df = pd.DataFrame(pca.fit_transform(train), columns=['PCA 1', 'PCA 2'])
firms = dfwide[(dfwide['year'] == 2020)]['Firm']
reduced_df['Firm'] = list(firms)

ax = reduced_df.plot(kind = 'scatter', x = ['PCA 1'], y = ['PCA 2'], figsize = (8,6))

plt.title('PCA dimensionality reduction')

for idx, row in reduced_df.iterrows():
    if (row['PCA 2'] > 5 or row['PCA 1'] > 10):
        ax.annotate(row['Firm'], (row['PCA 1'], row['PCA 2']),xytext=(5,-5),
                    textcoords='offset points', family='sans-serif', fontsize=9, color = '
```



DBSCAN

Density-based clustering methods, such as DBSCAN, identify clusters of data points by evaluating the proximity of one point to another. These methods typically employ Euclidean distance, a measure of the straight-line distance between two points, to determine cluster boundaries. A key parameter in DBSCAN clustering is epsilon, which defines the maximum distance between two cluster boundary points for them to be considered part of the same cluster.

While these clustering techniques are commonly employed to identify distinct groups of observations, our approach diverges by using them to detect outlier observations. We achieve this by determining the epsilon value that results in exactly 15 firms being unable to join any cluster. Additionally, I have set the min_samples parameter to 5, stipulating that a cluster must contain at least five observations to be recognized as such. This approach allows us to isolate outlier firms effectively, providing valuable insights into anomalies within the dataset.

See below the function I made to find these outliers.

```
def get_dbscan_outliers(df, calc_cols, number_of_outliers):
    #function that fins top X outliers using density-base clustering algorithm.
    outliers = []

    train = df[(df['year'] == 2020)][calc_cols]
    train = StandardScaler().fit_transform(train) #normalise data

    for dist in np.arange(0.1, 20, 0.1):

        db = DBSCAN(eps=dist, min_samples=5).fit(train) #outlier if there are less than 5

        dist_from_optimal = abs(len(pd.Series(db.labels_)[pd.Series(db.labels_ == -1)]) -

        outliers.append(dist_from_optimal) #i want 15 outliers

    optimal_dist = np.arange(0.1, 20, 0.1)[outliers.index(0)] # get dist that produces 15

    db = DBSCAN(eps=optimal_dist, min_samples=5).fit(train)

    table = df[(df['year'] == 2020)][['Firm']]

    table['db_group'] = db.labels_

    return table
```

Using DBSCAN in the way described above we find that the following 15 firms are outliers, in no particular order. Reassuringly, firms 4, 105, 166 and others are also outliers as identified in the previous visual inspection method, and the main isolation forest method used earlier in the report. As a final step, these top 5-10 firms should be passed on to supervisory teams, along with this report, for further investigation.

```
#get dbscan outliers
outliers = get_dbscan_outliers(dfwide, calc_cols, 15)
print(outliers[outliers['db_group'] == -1][['Firm']])
```

```
         Firm
9      Firm 10
39    Firm 105
154   Firm 127
544   Firm 199
614   Firm 210
884    Firm 26
```

```
944   Firm 270
1014  Firm 283
1134  Firm 304
1174  Firm 311
1304   Firm 34
1509   Firm 39
1844   Firm 52
2039   Firm 88
2099   Firm 99
```

Given the potentially very poor quality of data used in this report, the methods described here could also be used to detect instances of misreporting, and help flag these to data collection teams at the Bank.

Task 3

Introduction

Here I aim to detail the process and key factors involved in leveraging Azure cloud services to automate the generation and distribution of the above report. Our initial step is to define the product requirements, followed by a tailored Azure cloud solution to streamline the process effectively.

Defining the Product

The primary objective is to provide supervisors with periodic, automated insights on outlier insurance firms. To achieve this, standardization of the report format is crucial. This entails systematically extracting insights and findings through code, ensuring consistency and accuracy. A standardized set of charts and metrics could be ideal for this purpose. However, to add flexibility, transforming the product into a web-based application is recommended. This app would allow supervisors to interactively explore data and pinpoint outliers, offering a more hands-on approach to data analysis. The data, sourced quarterly from insurance firms for regulatory purposes, dictates a low data refresh frequency.

Therefore, the envisaged end product is a web-based application that identifies outlier insurance firms and updates every quarter, tailored to these specific requirements.

Key Considerations

In developing this Azure-based solution, several critical aspects must be considered:

Cost Management: Cloud services, while efficient, can escalate in cost rapidly. It's essential to operate within a pre-approved budget set by senior stakeholders. Opting for high-end, costly services may not be justified unless they offer significant value addition. Cost-effectiveness without compromising on essential features is the goal here.

Scalability: Given the project's nature, scalability is not a primary concern. The requirements, data volume, and refresh rate (quarterly) are expected to remain consistent over time. Therefore, the focus should be on selecting a solution that meets current needs without overemphasizing scalability.

Security, Compliance, and Data Governance: Ensuring the solution adheres to the bank's data governance rules and compliance standards, particularly when handling sensitive supervisory data, is paramount. Any proposed solution must be rigorously evaluated for its security measures and compliance with established protocols.

By addressing these considerations and focusing on a streamlined, cost-effective, and secure approach, the proposed Azure cloud service solution aims to enhance the efficiency and accessibility of report generation for supervisory purposes.

Azure pipeline

Given the above considerations, I should go for simple (and easy to use) solutions for the product data pipeline. It will look as follows:

1. **ADF**: Azure Data Factory is used to schedule data retrieval jobs on a weekly basis. Although the data is submitted quarterly, it may be revised more often than that, to account for this we refresh the data quarterly. I am also assuming that the data is available, possibly on a SQL server or Azure data storage service, which is accessible via azure data factory. This step includes some very basic data validation and transformation, with email alerts to devs in case of any issues.
2. **Azure SQL Database**: As the data is highly structured, I will use Azure SQL database for storage. This is a simple, easy-to-use solution which is highly suited to this type of product. The cost also scales with usage, which makes it a possibly cost-effective solution.
3. **Databricks**: Azure databricks is used to process data and fit the outlier detection models. These are computationally very inexpensive, due to small size of data and models, so this could potentially be moved to step four (posit connect) to save costs.
4. **Posit Connect**: The app is hosted on Posit Connect, which will connect to the Azure SQL Database. Alternatively, to stay within the Azure/Microsoft suite of products, the front-end could be made using Power BI.