# ocean_3_nov

November 3, 2022

[1]:
```
import pandas as pd
```

[2]:
```
dataset = pd.read_csv('scotland_dataset.csv')
```

[3]:
```
#inspecting the dataset
dataset.head()
```

[3]:
```
   Property_UPRN Postcode   POST_TOWN Date of Assessment  \
0   1.001101e+09  EH4 5EZ  EDINBURGH          01/01/2021
1   1.001951e+09  EH7 4HE  EDINBURGH          01/01/2021
2   1.000996e+09  EH4 2DL  EDINBURGH          02/01/2021
3   1.001257e+09  PH1 1SA      PERTH          02/01/2021
4   1.235709e+09  G78 1QN    Glasgow          02/01/2021

   Primary Energy Indicator (kWh/m²/year)  Total floor area (m²)  \
0                                   375.0                   94.0
1                                   250.0                  175.0
2                                   403.0                   72.0
3                                   174.0                   96.0
4                                   145.0                   58.0

   Current energy efficiency rating Current energy efficiency rating band  \
0                              53.0                                     E
1                              66.0                                     D
2                              61.0                                     D
3                              76.0                                     C
4                              79.0                                     C

   Potential Energy Efficiency Rating Potential energy efficiency rating band  \
0                                85.0                                       B
1                                80.0                                       C
2                                78.0                                       C
3                                87.0                                       B
4                                79.0                                       C

   …  Total current energy costs over 3 years (£)  \
0  …                                       3789.0
1  …                                       4635.0
```

```
2  …                                        3570.0
3  …                                        2049.0
4  …                                        1212.0

  Current heating costs over 3 years (£)  \
0                               2922.0
1                               4068.0
2                               2226.0
3                               1554.0
4                                828.0

   Potential heating costs over 3 years (£)  \
0                                  1548.0
1                                  3015.0
2                                  1191.0
3                                  1554.0
4                                   828.0

  Current hot water costs over 3 years (£)  \
0                                  645.0
1                                  246.0
2                                 1038.0
3                                  258.0
4                                  216.0

   Potential hot water costs over 3 years (£)  \
0                                     219.0
1                                     246.0
2                                     564.0
3                                     177.0
4                                     216.0

  Current lighting costs over 3 years (£)  \
0                                 222.0
1                                 321.0
2                                 306.0
3                                 237.0
4                                 168.0

  Potential lighting costs over 3 years (£) Part 1 Construction Age Band  \
0                                   222.0                   1930-1949
1                                   321.0                   1919-1929
2                                   207.0                   1965-1975
3                                   237.0                   1999-2002
4                                   168.0                 before 1919

    Built Form Property Type
```

```
0  Semi-Detached         House
1    End-Terrace         House
2  Semi-Detached          Flat
3    Mid-Terrace         House
4    Mid-Terrace          Flat

[5 rows x 48 columns]
```

[4]:
```
dataset.columns
```

[4]:
```
Index(['Property_UPRN', 'Postcode', 'POST_TOWN', 'Date of Assessment',
       'Primary Energy Indicator (kWh/m²/year)', 'Total floor area (m²)',
       'Current energy efficiency rating',
       'Current energy efficiency rating band',
       'Potential Energy Efficiency Rating',
       'Potential energy efficiency rating band',
       'Current Environmental Impact Rating',
       'Current Environmental Impact Rating Band',
       'Potential Environmental Impact Rating',
       'Potential Environmental Impact Rating Band',
       'CO2 Emissions Current Per Floor Area (kg.CO2/m²/yr)',
       'WALL_DESCRIPTION', 'WALL_ENERGY_EFF', 'ROOF_DESCRIPTION',
       'ROOF_ENERGY_EFF', 'FLOOR_DESCRIPTION', 'FLOOR_ENERGY_EFF',
       'FLOOR_ENV_EFF', 'WINDOWS_DESCRIPTION', 'WINDOWS_ENERGY_EFF',
       'WINDOWS_ENV_EFF', 'MAINHEAT_DESCRIPTION', 'MAINHEAT_ENERGY_EFF',
       'MAINHEAT_ENV_EFF', 'MAINHEATCONT_DESCRIPTION', 'MAINHEATC_ENERGY_EFF',
       'MAINHEATC_ENV_EFF', 'HOT_WATER_ENERGY_EFF', 'HOT_WATER_ENV_EFF',
       'LIGHTING_DESCRIPTION', 'LIGHTING_ENERGY_EFF', 'LIGHTING_ENV_EFF',
       'Current Emissions (T.CO2/yr)',
       'Potential Reduction in Emissions (T.CO2/yr)',
       'Total current energy costs over 3 years (£)',
       'Current heating costs over 3 years (£)',
       'Potential heating costs over 3 years (£)',
       'Current hot water costs over 3 years (£)',
       'Potential hot water costs over 3 years (£)',
       'Current lighting costs over 3 years (£)',
       'Potential lighting costs over 3 years (£)',
       'Part 1 Construction Age Band', 'Built Form', 'Property Type'],
      dtype='object')
```

[5]:
```
town_efficiency = dataset.groupby(['POST_TOWN'])['Current energy efficiency␣
 ↪rating'].mean()
```

[6]:
```
#noticing that the same town can appear multiple times
town_efficiency
```

[6]:
```
POST_TOWN
ABERDEEN         68.413096
ABERDEEN         82.000000
ABERDEENSHIRE    83.250000
ABERFELDY        63.464567
ABERLOUR         62.445122
                    …
Wick             70.650000
Wigtown          34.000000
Winchburgh       88.809524
Wishaw           76.875000
barlochan        39.000000
Name: Current energy efficiency rating, Length: 990, dtype: float64
```

[7]:
```
dataset.POST_TOWN.unique()
```

[7]:
```
array(['EDINBURGH ', 'PERTH ', 'Glasgow ', 'DOLLAR ', 'GLASGOW ',
       'LARBERT ', 'THORNHILL ', 'HADDINGTON ', 'DUNBAR ', 'DUNDEE ',
       'FORRES ', 'LEVEN ', 'CALLANDER ', 'KILWINNING ', 'PRESTWICK ',
       'INVERURIE ', 'ABERDEEN ', 'STONEHAVEN ', 'LONGNIDDRY ',
       'EAST LINTON ', 'GREENOCK ', 'BRODICK ', 'STRATHCARRON ',
       'ERSKINE ', 'SPEAN BRIDGE ', 'TRANENT ', 'LIVINGSTON ',
       'FRASERBURGH ', 'STORNOWAY ', 'NORTH BERWICK ', 'AUCHTERARDER ',
       'Aberdeen ', 'PETERHEAD ', 'COATBRIDGE ', 'BANCHORY ', 'BATHGATE ',
       'Laurencekirk ', 'Edinburgh ', 'Dalkeith ', 'PAISLEY ',
       'DUNFERMLINE ', 'ROSLIN ', 'PENICUIK ', 'BERWICK UPON TWEED ',
       'ELGIN ', 'GLENROTHES ', 'DUMBARTON ', 'CUPAR ', 'WISHAW ',
       'MUSSELBURGH ', 'DENNY ', 'KILMARNOCK ', 'FALKIRK ', 'WESTHILL ',
       'GIRVAN ', 'STIRLING ', 'JOHNSTONE ', 'ELLON ', 'AYR ', 'TROON ',
       'KIRKCALDY ', 'GOREBRIDGE ', 'ARBROATH ', 'KINROSS ', 'HAMILTON ',
       'CLYDEBANK ', 'INVERNESS ', 'AIRDRIE ', 'FOCHABERS ',
       'HELENSBURGH ', 'DUMFRIES ', 'DUNBLANE ', 'LOSSIEMOUTH ',
       'KIRRIEMUIR ', 'KILMACOLM ', 'MOTHERWELL ', 'BIGGAR ', 'DUNOON ',
       'GOUROCK ', 'BEITH ', "BO'NESS ", 'BLAIRGOWRIE ', 'CURRIE ',
       'PRESTONPANS ', 'BONNYBRIDGE ', 'TILLICOULTRY ', 'WEST KILBRIDE ',
       'CRIEFF ', 'RENFREW ', 'GALSTON ', 'DALKEITH ', nan, 'LARKHALL ',
       'SHOTTS ', 'IRVINE ', 'East Lothian ', 'Fife ', 'GRANGEMOUTH ',
       'KIRKCUDBRIGHT ', 'TIGHNABRUAICH ', 'Angus ', 'Inverclyde ',
       'BROXBURN ', 'HUMBIE ', 'KEITH ', 'BONNYRIGG ', 'Invergordon ',
       'NEWMILNS ', 'ALLOA ', 'ALEXANDRIA ', 'OBAN ', 'LINLITHGOW ',
       'ST ANDREWS ', 'MENSTRIE ', 'INNERLEITHEN ', 'CUMNOCK ',
       'MONTROSE ', 'ULLAPOOL ', 'FORFAR ', 'CASTLE DOUGLAS ',
       'ANSTRUTHER ', 'LANARK ', 'HUNTLY ', 'TAIN ', 'MILLTIMBER ',
       'COWDENBEATH ', 'TAYPORT ', 'LOCHGILPHEAD ', 'ANNAN ', 'ABERLOUR ',
       'JEDBURGH ', 'BELLSHILL ', 'BRECHIN ', 'BUCKIE ', 'ALFORD ',
       'BANFF ', 'DINGWALL ', 'Inverness ', 'ISLE OF LEWIS ',
       'Musselburgh ', 'Hamilton ', 'MUIR OF ORD ', 'Midlothian ',
```

'BURNTISLAND ', 'Ormiston ', 'BISHOPTON ', 'SOUTH QUEENSFERRY ',
'PORTREE ', 'North Ayrshire ', 'Montrose ', 'LAUDER ',
'STEVENSTON ', 'DORNOCH ', 'HERIOT ', 'KIRKLISTON ', 'PEEBLES ',
'LOCHGELLY ', 'AVIEMORE ', 'ROY BRIDGE ', 'CROMARTY ',
'SALTCOATS ', 'NEWPORT ON TAY ', 'FORT WILLIAM ', 'CARLUKE ',
'ISLE OF BUTE ', 'DUNS ', 'LARGS ', 'STRANRAER ', 'KIRKWALL ',
'INVERGORDON ', 'HALKIRK ', 'WICK ', 'GALASHIELS ', 'EYEMOUTH ',
'LANGHOLM ', 'LOCHWINNOCH ', 'NEWTON STEWART ', 'SHETLAND ',
'KELTY ', 'ORKNEY ', 'St Andrews ', 'Gullane ', 'FORTROSE ',
'Penicuik ', 'Muirkirk ', 'Kirkcaldy ', 'Coatbridge ', 'Morvern ',
'Duns ', 'Isle of Harris ', 'SANQUHAR ', 'Falkirk ', 'Orkney ',
'Tain ', 'ABOYNE ', 'ARDROSSAN ', 'BEAULY ', 'KILBIRNIE ',
'SELKIRK ', 'STRATHAVEN ', 'HAWICK ', 'INVERKEITHING ', 'DUNKELD ',
'LOANHEAD ', 'NAIRN ', 'PORT GLASGOW ', 'COLDSTREAM ', 'BALERNO ',
'DARVEL ', 'WEST CALDER ', 'LAIRG ', 'Larkhall ', 'Haddington ',
'Banff ', 'Johnstone ', 'Bonnyrigg ', 'Dundee ', 'Roslin ',
'West Calder ', 'Gorebridge ', 'Poolewe ', 'Mauchline ',
'Isle of Lewis ', 'Kilmarnock ', 'Cupar ', 'Isle of Skye ',
'Shotts ', 'DALBEATTIE ', 'ABERFELDY ', 'CAMPBELTOWN ', 'DALRY ',
'LAURENCEKIRK ', 'BRIDGE OF WEIR ', 'MACDUFF ', 'PITLOCHRY ',
'LOCKERBIE ', 'ARDGAY ', 'Kingseat ', 'MAUCHLINE ', 'Alexandria ',
'Clackmannanshire ', 'East Kilbride ', 'Robroyston ', 'Keith ',
'Ardrossan ', 'Stirling ', 'Annan ', 'ISLE OF SKYE ',
'ACHNASHEEN ', 'KELSO ', 'CAIRNDOW ', 'TURRIFF ', 'MELROSE ',
'MAYBOLE ', 'TARBERT ', 'LASSWADE ', 'CARNOUSTIE ',
'ISLE OF HARRIS ', 'MILLPORT ', 'WALKERBURN ', 'Ayrshire ',
'Wick ', 'ALVA ', 'Twechar ', 'Bucksburn ', 'Clydebank ',
'Isle of Arran ', 'Spean Bridge ', 'Isle Of Arran ',
'ISLE OF NORTH UIST ', 'GULLANE ', 'KYLE ', 'DOUNE ', 'ACHARACLE ',
'STROMNESS ', 'NEWBRIDGE ', 'Saltcoats ', 'Bilston ', 'Leven ',
'GORDON ', 'Newton Mearns ', 'INVERKIP ', 'East Renfrewshire ',
'APPIN ', 'Livingston ', 'Catrine ', 'Newton Stewart ',
'ARROCHAR ', 'EARLSTON ', 'PETERCULTER ', 'JUNIPER GREEN ',
'AVOCH ', 'GRETNA ', 'Kincraig ', 'Troon ', 'Whitburn ',
'Blackridge ', 'Leven, Fife ', 'CARRBRIDGE ', 'GOLSPIE ',
'Uddingston ', 'Castle Douglas ', 'Fort William ', 'MINTLAW ',
'Scottish Borders ', 'KIRKNEWTON ', 'Barrhead ', 'DALMALLY ',
'Milltimber ', 'Irvine ', 'Seamill ', 'Isle of Bute ', 'INSCH ',
'CLACKMANNAN ', 'SKELMORLIE ', 'ALNESS ', 'Largs ', 'Beauly ',
'Forfar ', 'West Linton ', 'Brechin ', 'PLOCKTON ',
'ISLE OF MULL ', 'ISLE OF TIREE ', 'ISLE OF ARRAN ',
'ISLE OF BENBECULA ', 'Cumnock ', 'MOFFAT ', 'Tillicoultry ',
'Galashiels ', 'THURSO ', 'Muir of Ord ', 'Dumfries ',
'EAST RENFREWSHIRE ', 'Kinross-shire ', 'KILLIN ', 'Shetland ',
'Airdrie ', 'Bishopton ', 'Cruden Bay ', 'Banchory ', 'Oban ',
'Lumphanan ', 'Aberdeenshire ', 'GARGUNNOCK ', 'Kilbirnie ',
'Hollybush ', 'CANONBIE ', 'WEST LINTON ', 'Portpatrick ',

'Kilbarchan ', 'NEWTONMORE ', 'Inverurie ', 'Dalmally ', 'Perth ',
'GRANTOWN-ON-SPEY ', 'GARVE ', 'Cumbernauld ', 'By Thurso ',
'DUMFRIESSHIRE ', 'Govan ', 'Inverness  ', 'Nairn ', 'Loanhead ',
'Kinloss ', 'KINLOCHLEVEN ', 'Renfrew ', 'Dunfermline ',
'ISLE OF SOUTH UIST ', 'INVERARAY ', 'KINGUSSIE ', 'NETHY BRIDGE ',
'Elgin ', 'Dingwall ', 'Motherwell ', 'Lauder ', 'Aberfoyle ',
'Dumbarton ', 'ISLE OF ISLAY ', 'ROGART ', 'HELMSDALE ',
'LOCHAILORT ', 'GLENFINNAN ', 'Chryston ', 'Ellon ',
'Sma Glen, Crieff ', 'Portree ', 'Canonbie ', 'North Berwick ',
'COCKBURNSPATH ', 'STRATHPEFFER ', 'Cambuslang ', 'Ayr ',
'ISLE OF BARRA ', 'Dunbar ', 'Kyle of Lochalsh ', 'Dufftown ',
'Inzevar ', 'Linlithgow ', 'Peebles ', 'BOAT OF GARTEN ',
'WHITBURN ', 'Thornhill ', 'Bridge Of Allan ', 'Inverness-shire ',
'Carluke ', 'Kirriemuir ', 'Paisley ', 'Dunkeld ', 'MUNLOCHY ',
'BRORA ', 'TAYNUILT ', 'NEWCASTLETON ', 'Lanark ', 'Lochgelly ',
'Darvel ', 'North Uist ', 'Lossiemouth ', 'ARISAIG ', 'Kelso ',
'Isle Of North Uist ', 'LYBSTER ', 'BIshopton ', 'Bellshill ',
'Broadford ', 'MALLAIG ', 'BALLATER ', 'Helensburgh ',
'Glasgow City ', 'Aviemore ', 'Ballachulish ', 'Oxton ',
'BALLACHULISH ', 'Broxburn ', 'Ferniegair ', 'ROSEWELL ',
'Strontian ', 'Garve ', 'Camelon ', 'Anstruther ', 'GAIRLOCH ',
'Cumbernauld  ', 'CLACKMANNANSHIRE ', 'Isle Of Lewis ',
'Drumnadrochit ', 'Newmilns ', 'Blair Atholl ', 'Gairloch ',
'Dalbeattie ', 'GRANTOWN ON SPEY ', 'QUARRIERS VILLAGE ',
'Coldstream ', 'Gordon ', 'Stewarton ', 'Invergarry ', 'Crieff ',
'Gatehead ', 'WEMYSS BAY ', 'BALLINDALLOCH ', 'LOCHEARNHEAD ',
'NEWTON MEARNS ', 'South Queensferry ', 'Fairlie ', 'Skelmorlie ',
'Fintry ', 'Winchburgh ', 'Stranraer ', 'East Ayrshire ',
'Prestwick ', 'BISHOPBRIGGS ', 'Lenzie ', 'Erskine ', 'STRATHDON ',
'Johnston ', 'East Calder ', 'Cowdenbeath ', 'Isles of Lewis ',
'Prestonpans ', 'Wishaw ', 'Kingussie ', 'Lochwinnoch ',
'Lockerbie ', 'Fortrose ', 'Brora ', 'Auchterarder ', 'Hopeman ',
'Lasswade ', 'Armadale ', 'Carnoustie ', 'Sutherland ',
'NEWTOWN ST BOSWELL ', 'NEWTON ST BOSWELLS ', 'COLINTRAIVE ',
'Udny ', 'Granton On Spey ', 'Ross-shire ', 'Dunblane ', 'Hawick ',
'Kilmaurs ', 'Tranent ', 'PATHHEAD ', 'Aboyne ', 'Dollar ',
'Isle of Arran  ', 'Kyle ', 'Strome Ferry ', 'Pitlochry ',
'Alness ', 'FORT AUGUSTUS ', 'ISLE OF COLL ', 'Alloa ', 'Alva ',
'Newtongrange ', 'Dunoon ', 'Penpont ', 'CRIANLARICH ',
'Crainlarich ', 'Symington ', 'Bo’ness ', 'Callander ', 'Thurso ',
'Polbeth ', 'Auchinleck ', 'Stonehaven ', 'Bonar Bridge ',
'BERRIEDALE ', 'Biggar ', 'Arbroath ', 'Western Isles ', 'Kemnay ',
'Beeswing ', 'By Maybole ', 'Strathcarron ', 'Greenock ',
'Isle of North Uist ', 'Innerleithen ', 'ISLE OF COLONSAY ',
'Reddingmuirhead ', 'Maybole ', 'Cullen ', 'Alves ', 'Inveraray ',
'ARDERSIER ', 'Kilwinning ', 'Ancrum ', 'Auldearn ',
'Isle of Mull ', 'Doune ', 'Newtonmore ', 'Hyndland ', 'Dunlop ',

'Longniddry ', 'Crocketford ', 'Huntly ', 'Fort Augustus ',
'Dumfries and Galloway ', 'Dalmellington ', 'Strathpeffer ',
'East Linton ', 'Port Logan ', 'Glentress ', 'West Lothian ',
'Portsonachan ', 'DUNBEATH ', 'Dores ', 'Aberlour ', 'Moray ',
'Forres ', 'Garmouth ', 'Isle Of Skye ', 'Heiton ',
'CASLTE DOUGLAS ', 'Eaglesham ', 'Archiestown ', 'Denny ',
'Loch Katrine ', 'Burrelton ', 'Grantown-On-Spey ', 'MACHLINE ',
'NORTH LANARKSHIRE ', 'Glenboig ', 'Lairg ', 'Rosewell ',
'CARNWATH ', 'By Forfar ', 'Mlliport ', 'Glenrothes ', 'Tarves ',
'BRAIDWOOD ', 'St Ola ', 'KIRKINTILLOCH ', 'Melrose ',
'Grantown-on-Spey ', 'Gorebridge  ', 'Crookston ', 'Dunragit ',
'Caithness ', 'Lochinver ', 'Burghead ', 'Bathgate ', 'Dunipace ',
'Bridge of Allan ', 'Eyemouth ', 'Fardalehill ', 'ABERNETHY ',
'Nr Ballantrae ', 'St Boswells ', 'Peterculter ', 'Shiskine ',
'Rosemarkie ', 'Blairgowrie ', 'Banffshire ', 'Peterhead ',
'ISLE OF SCALPAY ', 'Lugar ', 'Kirkcudbright ', 'Ballindalloch ',
'Fraserburgh ', 'Kirkcolm ', 'CARSTAIRS JUNCTION ',
'Boat of Garten ', 'ISLE OF IONA ', 'Dumfriesshire ',
'South Ayrshire ', 'Strathblane ', 'Ullapool ', 'Highland ',
'Stevenston ', 'Barcaldine ', 'Turriff ', 'Acharacle ',
'PEEBLESSHIRE ', 'Strathdon ', 'Drummore ', 'Millport ',
'Crosshill ', 'Aberfeldy ', 'Kincardine ', 'WALLYFORD ',
'Garnethill ', 'Strathnairn ', 'KINBRACE ', 'Berwickshire ',
'Port Glasgow ', 'Coldingham ', 'Glasgow  ', 'Muirhead ',
'CATRINE ', 'Sanquhar ', 'Muckhart ', 'Dornoch ', 'Newcraighall ',
'Meikleour ', 'Edinburgh  ', 'Glenmavis ', 'Annbank ', 'Tiree ',
'West Plean ', 'DALWHINNIE ', 'Invergodon ', 'Kirkintilloch ',
'Isle of Benbecula ', 'Campbeltown ', 'Tealing ', 'Kirkcubright ',
'Grangemouth ', 'Balerno ', 'Rogart ', 'Mallaig ', 'Dundonnell ',
'Portsoy ', 'Bridge of Weir ', 'Pollock ', 'Ardgay ', 'Gartly ',
'West Kilbride ', 'ABERDEENSHIRE ', 'Gardenstown ', 'Wigtown ',
'Dunning ', 'Ormitston ', 'Crossgates ', 'Girvan ', 'Kinross ',
'Galston ', 'Nigg ', 'Mount Vernon ', 'LANARKSHIRE ',
'Stonehouse ', 'Coalsnaughton ', 'Newton Merans ', 'Boness ',
'Ballater ', 'Avoch ', 'Avonbridge ', 'Lanarkshire ', 'Earlston ',
'Stormness ', 'Roybridge ', 'Rothesay ', 'Carrbridge ',
'SOUTH QUENNSFERRY ', 'Crosshouse ', 'Brechin, Angus ', 'Argyll ',
'Killin ', 'Achnasheen ', 'Torrance ', 'Fochabers ', 'Glencoe ',
'Gourock ', 'Strathaven ', 'LATHERON ', 'Walkerburn ', 'Pathhead ',
'Kirkliston ', 'Holytown ', 'Skene ', 'Isle of South Uist ',
'Arrochar ', 'Taynuilt ', 'Cromdale ', 'Monkton ', 'Fearn ',
'ISLE OF GIGHA ', 'Coupar Angus ', 'Errol ', 'GLENTROOL ',
'Fort William  ', 'Lochmaben  ', 'Gattonside ', 'GARTOCHARN ',
'Brightons ', 'Nr Callander ', 'Evanton ', 'Berwick Upon Tweed ',
'Kingswells ', 'INVERGARRY ', 'Newbridge ', 'Dairy ',
'Tighnabruaich ', 'Isle of Coll ', 'Kiltarlity ', 'Isle of Islay ',
'STROME FERRY ', 'Dalry ', 'Newport on Tay ', 'Glentromie ',

```
'Neilston ', 'Hurlford ', 'Bridge of Don ', 'South Lanarkshire ',
'Myrehead ', 'Satlcoats ', 'Stoneykirk ', 'Inchinnan ',
'Kirkwall ', 'North Connel ', 'Stepps ', 'Philpstoun ',
'Grantown On Spey ', 'STEWARTON ', 'Ross-Shire ', 'Inverbervie ',
'Kinlochleven ', 'Lesmahagow ', 'Sandend ', 'East of Lindores ',
'Tarbolton ', 'Mussleburgh ', 'Darnick ', 'New Cumnock ',
'Isle of Cumbrae ', 'Rhynie ', 'Insch ', "Bo'ness ",
'Eskdalemuir  ', 'Old Rayne ', 'ST FERGUS ', 'Corsock ',
'Castletown ', 'Selkirk ', 'Newburgh ', 'Granton on Spey ',
'Sanday ', 'Humbie ', 'Maryhill ', 'Bannockburn ',
'Newport-on-Tay ', 'Milngavie ', 'Ardrishaig ', 'Cairndow ',
'ISLE OF JURA ', 'FAIRLIE ', 'Lochbroom ', 'Dalrymple ', 'Beith ',
'North Aryshire ', 'Nethy Bridge ', 'Kinbrace ', 'East Hardgate ',
'Morebattle ', 'Bettyhill ', 'Kirkholm ', 'Kelton ', 'Bearsden ',
'Halkirk ', 'Auchtermuchty ', 'Pencaitland ', 'Athelstaneford ',
'Durris ', 'Kirkpatrick Durham ', 'Morayshire ', 'Kilcreggan ',
'Kilsyth ', 'Cromarty ', 'Shetland Islands ', 'Dumfries  ',
'Grantown on Spey ', 'Cockburnspath ', 'PAXTON ', 'Bonawe ',
'Kyleakin ', 'Findhorn ', 'Golspie ', 'Comrie ', 'Polmont ',
'Lochgilphead ', 'New Stevenston ', 'Roseisle ', 'Tayport ',
'Lonmay ', 'Muir Of Ord ', 'Maryburgh ', 'Park ', 'Kilmartin ',
'Bishopbriggs ', 'Stornaway ', 'Campbe;town ', 'Buckie ',
'Renfrewshire ', 'South Aryshire ', 'Stirlingshire ',
'Cellardyke ', 'Laggan ', 'West Dunbartonshire ', 'Stornoway ',
'Knoydart ', 'Heriot ', 'Wallyford ', 'Fortingall ', 'Carrmyllie ',
'Jedburgh ', 'FORSINARD ', 'MOSSBLOWN ', 'Gatehouse of Fleet ',
'Alford ', 'Dysart ', 'Newmachar ', 'Auchinleck  ', 'Perthshire ',
'Granton-On-Spey ', 'Mugiemoss ', 'Methil ', 'Kilwinning  ',
'South Ayrshire  ', 'Dunbartonshire ', 'Newcastleton ', 'Larbert ',
'Kilmacolm ', 'Applecross ', 'Lochailort ', 'Blantyre ',
'Kirkbean ', 'MACHLINE  ', 'MIDLOTHIAN ', 'Kintore ', 'Coylton ',
'Lumphanan  ', 'Forsinard ', 'ARGYLL AND BUTE ', 'Gorthleck ',
'MACMERRY ', 'Westhill ', 'Pluscarden ', 'Upper Largo ',
'Parkgate ', 'Duror ', 'Carnbroe ', 'Argyll Street ',
'WINCHBURGH ', 'Clarkston ', 'ROXBURGHSHIRE ', 'Mossblown ',
'Wemyss Bay ', 'Blackford ', 'North Queensferry ', 'Monkton  ',
'Langholm ', 'Drongan ', 'Caldercruix ', 'Strachur ', 'Appin ',
'Burray ', 'Edzell ', 'EAST LOTHIAN ', 'Blairs ', 'Moffat ',
'Borgue ', 'Latheron ', 'Lennoxtown ', 'Dunbeath ', 'Auldgirth ',
'Kinross, Fife ', 'Ballantrae ', 'BRIDGE OF ORCHY ',
'Argyll &amp; Bute ', 'ISLE OF EIGG ', 'Perth &amp; Kinross ',
'CLACHAN OF CAMPSIE ', 'Thornton ', 'Glassford ', 'Dundonald ',
'Staffin ', 'Gretna ', 'Broughty Ferry ', 'INVERNESS  ', 'FIFE ',
'Boat Of Garten ', 'Udny Station ', 'Glendaruel ', 'Balmullo ',
'Rathen ', 'Stuartfield ', 'Ochiltree ', 'Daviot ', 'Reay ',
'North Lanarkshire ', 'Isle of Scalpy ', 'Craigellachie ',
'Banknock ', 'Cumbrae ', 'Drumndadrochit ', 'Giffnock ',
```

```
'BLINDWELLS ', 'By Ayr ', 'Glengarnock ', 'Orkney  ', 'Meigle ',
'Berriedale ', 'Strathcarron   ', 'Isle Of South Uist ',
'Uplawmoor ', 'North  Middleton ', 'Firth ',
'East Dunbartonshire ', 'Lower Foyers ', 'Acharcle ', 'OXTON ',
'West Post ', 'Cardross ', 'EAST RENFREWSHIRE  ', 'Drymen ',
'ARGYLL ', 'Bridge Of  Weir Road ', 'Malaig ', 'TROON  ',
'Uddingston  ', 'ISLE OF ARRAN  ', 'Brighouse Bay  ',
'barlochan  ', 'Guardbridge ', 'COUSLAND  ', 'Edingburgh ',
'ABERLOUR  ', 'DUNDONALD ', 'Kilmarnock  ', 'Road BIshopton ',
'Ardfern ', 'Bathgate  ', 'Arbroath  ', 'Saltcoats  ', 'Inverkip ',
'Innerleithen  ', 'Pumpherston ', 'ABERDEEN  ',
'Quarriers Village ', 'Ayton ', 'ROTHIENORMAN ', 'Sorn ',
'Lochearnhead '], dtype=object)
```

[8]:
```
len(dataset.POST_TOWN.unique())
```

[8]:
```
991
```

[9]:
```
dataset['POST_TOWN'] = dataset.POST_TOWN.astype(str).str.lower()
```

[10]:
```
len(dataset.POST_TOWN.unique())
```

[10]:
```
692
```

[11]:
```
dataset['POST_TOWN'] = dataset['POST_TOWN'].str.
 ↪replace('edingburgh','edinburgh')
```

[12]:
```
len(dataset.POST_TOWN.unique())
```

[12]:
```
691
```

[13]:
```
dataset.isnull().sum()
```

[13]:
```
Property_UPRN                                           0
Postcode                                                0
POST_TOWN                                               0
Date of Assessment                                      0
Primary Energy Indicator (kWh/m²/year)                  0
Total floor area (m²)                                   0
Current energy efficiency rating                        0
Current energy efficiency rating band                   0
Potential Energy Efficiency Rating                      0
Potential energy efficiency rating band                 0
Current Environmental Impact Rating                     0
Current Environmental Impact Rating Band                0
Potential Environmental Impact Rating                   0
Potential Environmental Impact Rating Band              0
CO2 Emissions Current Per Floor Area (kg.CO2/m²/yr)     0
```

```
WALL_DESCRIPTION                                         0
WALL_ENERGY_EFF                                          0
ROOF_DESCRIPTION                                         0
ROOF_ENERGY_EFF                                      39235
FLOOR_DESCRIPTION                                        0
FLOOR_ENERGY_EFF                                    117202
FLOOR_ENV_EFF                                       117202
WINDOWS_DESCRIPTION                                      0
WINDOWS_ENERGY_EFF                                       0
WINDOWS_ENV_EFF                                          0
MAINHEAT_DESCRIPTION                                     0
MAINHEAT_ENERGY_EFF                                      0
MAINHEAT_ENV_EFF                                         0
MAINHEATCONT_DESCRIPTION                                 0
MAINHEATC_ENERGY_EFF                                     0
MAINHEATC_ENV_EFF                                        0
HOT_WATER_ENERGY_EFF                                     0
HOT_WATER_ENV_EFF                                        0
LIGHTING_DESCRIPTION                                     0
LIGHTING_ENERGY_EFF                                      0
LIGHTING_ENV_EFF                                         0
Current Emissions (T.CO2/yr)                             0
Potential Reduction in Emissions (T.CO2/yr)              0
Total current energy costs over 3 years (£)              0
Current heating costs over 3 years (£)                   0
Potential heating costs over 3 years (£)                 0
Current hot water costs over 3 years (£)                 0
Potential hot water costs over 3 years (£)               0
Current lighting costs over 3 years (£)                  0
Potential lighting costs over 3 years (£)                0
Part 1 Construction Age Band                         29972
Built Form                                             622
Property Type                                            0
dtype: int64
```

[14]:
```
town_efficiency = dataset.groupby(['POST_TOWN'])['Current energy efficiency␣
↪rating'].mean()
```

[15]:
```
dataset['POST_TOWN'] = dataset['POST_TOWN'].str.rstrip()
```

[16]:
```
len(dataset.POST_TOWN.unique())
```

[16]: 665

# 1 Part 1: Rankings

## 1.1 1 Rank Towns by Current Energy Efficiency Rating

[17]:
```
town_efficiency = dataset.groupby(['POST_TOWN'])['Current energy efficiency␣
↪rating'].mean().reset_index(name='Mean')
town_efficiency = town_efficiency.sort_values(by='Mean', ascending=False ).
↪reset_index(drop=True)
town_efficiency.head(5)
```

[17]:
```
            POST_TOWN        Mean
0          gartocharn  115.000000
1         bannockburn  111.750000
2  gatehouse of fleet  101.000000
3    north lanarkshire  97.117647
4       south aryshire  96.000000
```

## 1.2 2 Rank Towns by potential energy efficiency rating

[18]:
```
town_potential_eff = dataset.groupby(['POST_TOWN'])['Potential Energy␣
↪Efficiency Rating'].mean().reset_index(name='Mean')
town_potential_eff=  town_potential_eff.sort_values(by='Mean', ascending=False).
↪reset_index(drop=True)
town_potential_eff.head(5)
```

[18]:
```
            POST_TOWN   Mean
0  gatehouse of fleet  128.0
1              comrie  123.0
2              sanday  119.0
3         crainlarich  118.5
4          gartocharn  118.0
```

## 1.3 3 Rank Towns by current environmental impact rating

[19]:
```
town_curr_env_impact = dataset.groupby(['POST_TOWN'])['Current Environmental␣
↪Impact Rating'].mean().reset_index(name='Mean')
town_curr_env_impact = town_curr_env_impact.sort_values(by='Mean',␣
↪ascending=False).reset_index(drop=True)
town_curr_env_impact.head(5)
```

[19]:
```
           POST_TOWN        Mean
0         gartocharn  113.000000
1        bannockburn  109.750000
2  north lanarkshire   98.705882
3     south aryshire   97.000000
4            ardfern   96.000000
```

## 1.4 3 … and note if there have been periods where houses were more or less environmentally friendly

[20]:
```
town_env_date = dataset.groupby(['POST_TOWN', 'Date of Assessment' ])['Current␣
↪Environmental Impact Rating'].mean().reset_index(name='Mean')
town_env_date
```

[20]:
```
      POST_TOWN Date of Assessment       Mean
0      aberdeen         01/02/2021  64.818182
1      aberdeen         01/03/2021  70.153846
2      aberdeen         01/04/2021  69.333333
3      aberdeen         01/06/2021  72.755556
4      aberdeen         01/07/2021  68.844444
…           …                …          …
36932    wishaw         30/09/2021  70.000000
36933    wishaw         30/11/2021  45.000000
36934    wishaw         31/03/2021  67.200000
36935    wishaw         31/05/2021  68.750000
36936    wishaw         31/08/2021  72.250000

[36937 rows x 3 columns]
```

[21]:
```
town_env_date.dtypes
```

[21]:
```
POST_TOWN              object
Date of Assessment     object
Mean                  float64
dtype: object
```

[22]:
```
town_env_date['Date of Assessment'] = pd.to_datetime(town_env_date['Date of␣
↪Assessment'], format='%d/%m/%Y')
```

[23]:
```
town_env_date.dtypes
```

[23]:
```
POST_TOWN                     object
Date of Assessment    datetime64[ns]
Mean                         float64
dtype: object
```

[24]:
```
import matplotlib.pyplot as plt
```

[25]:
```
gartocharn = town_env_date[town_env_date['POST_TOWN']=='gartocharn']
gartocharn
```

[25]:
```
        POST_TOWN Date of Assessment   Mean
15324  gartocharn         2021-05-26  113.0
```

[26]:
```
bannockburn = town_env_date[town_env_date['POST_TOWN']=='bannockburn']
bannockburn
```

[26]:
```
        POST_TOWN Date of Assessment    Mean
3803  bannockburn         2021-06-28  109.75
```

[27]:
```
gatehouse_of_fleet = town_env_date[town_env_date['POST_TOWN']=='gatehouse of␣
↪fleet']
gatehouse_of_fleet
```

[27]:
```
                POST_TOWN Date of Assessment  Mean
15348  gatehouse of fleet         2021-08-19  83.0
```

[28]:
```
north_lanarkshire = town_env_date[town_env_date['POST_TOWN']=='north␣
↪lanarkshire']
north_lanarkshire
```

[28]:
```
               POST_TOWN Date of Assessment     Mean
28915  north lanarkshire         2021-11-11  99.1875
28916  north lanarkshire         2021-03-25  91.0000
```

[29]:
```
south_aryshire = town_env_date[town_env_date['POST_TOWN']=='south aryshire']
south_aryshire
```

[29]:
```
            POST_TOWN Date of Assessment  Mean
32811  south aryshire         2021-08-04  97.0
```

[30]:
```
ardfern = town_env_date[town_env_date['POST_TOWN']=='ardfern']
ardfern
```

[30]:
```
     POST_TOWN Date of Assessment  Mean
2354   ardfern         2021-12-14  96.0
```

[31]:
```
measurements = dataset.groupby(['POST_TOWN'])['POST_TOWN'].count().
↪reset_index(name='count')
```

[32]:
```
measurements = measurements.sort_values(by='count', ascending=False).
↪reset_index(drop=True)
measurements
```

[32]:
```
           POST_TOWN  count
0            glasgow  37529
1          edinburgh  19276
2           aberdeen   9226
3             dundee   5782
4            paisley   3136
..               …      …
660         myrehead      1
```

```
661  bridge of orchy      1
662          neilston      1
663   new stevenston      1
664     lower foyers      1

[665 rows x 2 columns]
```

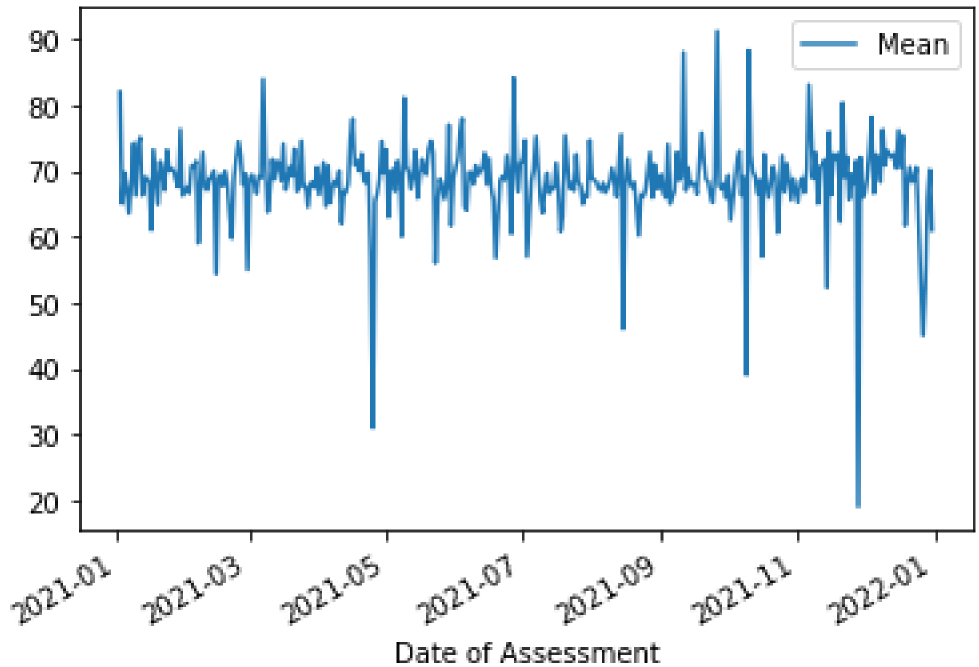
[33]:
```
glasgow = town_env_date[town_env_date['POST_TOWN']=='glasgow']
glasgow
```

[33]:
```
      POST_TOWN Date of Assessment       Mean
15486   glasgow         2021-02-01  67.567164
15487   glasgow         2021-03-01  68.902256
15488   glasgow         2021-04-01  66.358779
15489   glasgow         2021-05-01  73.200000
15490   glasgow         2021-06-01  70.469613
…          …                 …          …
15835   glasgow         2021-03-31  70.516129
15836   glasgow         2021-05-31  68.906250
15837   glasgow         2021-07-31  74.642857
15838   glasgow         2021-08-31  67.344262
15839   glasgow         2021-12-31  61.000000

[354 rows x 3 columns]
```

[34]:
```
#Glasgow does not really show a trend.
glasgow.plot(x='Date of Assessment', y='Mean')
```

[34]:
```
<AxesSubplot:xlabel='Date of Assessment'>
```




[35]:
```
edinburgh = town_env_date[town_env_date['POST_TOWN']=='edinburgh']
edinburgh
```

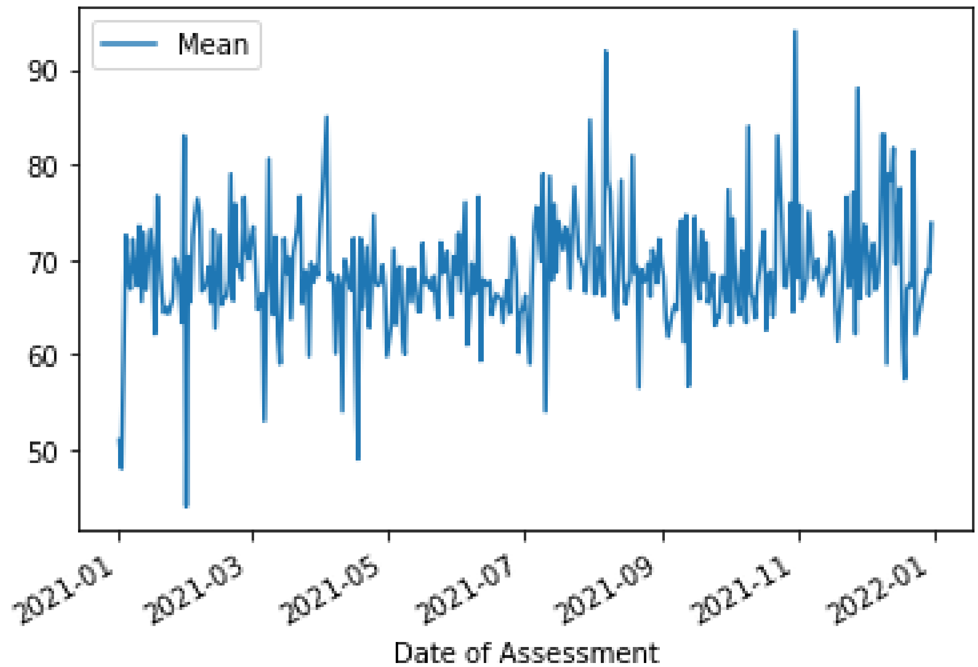
[35]:

```
        POST_TOWN Date of Assessment       Mean
12489   edinburgh         2021-01-01  51.000000
12490   edinburgh         2021-02-01  70.355932
12491   edinburgh         2021-03-01  72.357143
12492   edinburgh         2021-04-01  73.457447
12493   edinburgh         2021-05-01  59.800000
…             …                  …        …
12829   edinburgh         2021-05-31  70.320000
12830   edinburgh         2021-07-31  84.666667
12831   edinburgh         2021-08-31  72.105882
12832   edinburgh         2021-10-31  94.000000
12833   edinburgh         2021-12-31  73.800000

[345 rows x 3 columns]
```

[36]:
```
#Edinburgh seems to show larger impact at the end of the year
edinburgh.plot(x='Date of Assessment', y='Mean')
```

[36]: <AxesSubplot:xlabel='Date of Assessment'>



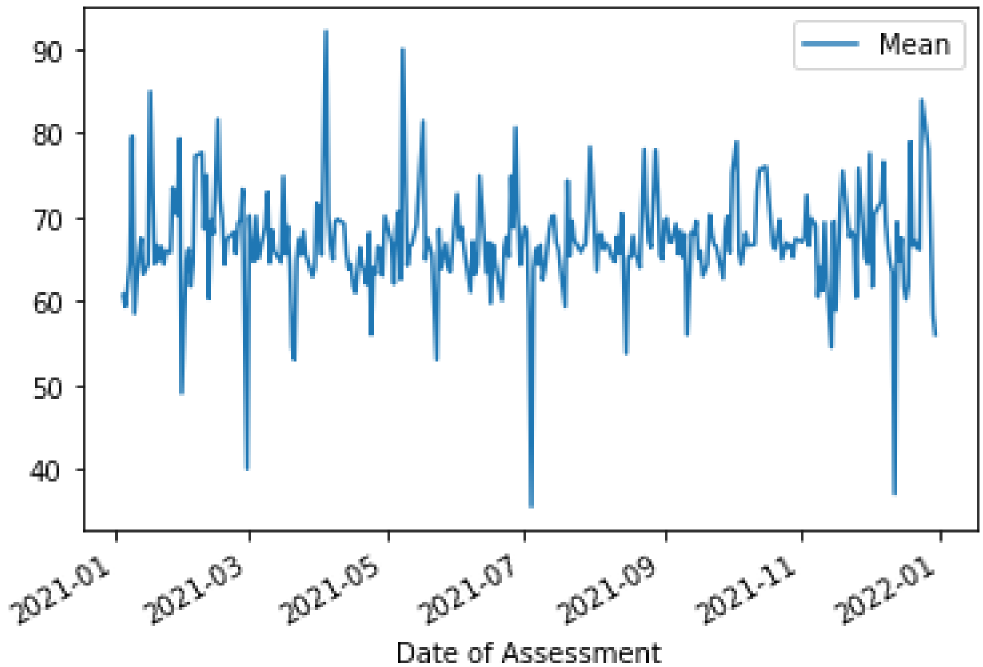

[37]:
```
aberdeen = town_env_date[town_env_date['POST_TOWN']=='aberdeen']
aberdeen
```

[37]:
```
    POST_TOWN Date of Assessment       Mean
0    aberdeen         2021-02-01  64.818182
1    aberdeen         2021-03-01  70.153846
2    aberdeen         2021-04-01  69.333333
3    aberdeen         2021-06-01  72.755556
4    aberdeen         2021-07-01  68.844444
..        …                  …        …
290  aberdeen         2021-11-30  64.500000
291  aberdeen         2021-12-30  56.000000
292  aberdeen         2021-03-31  71.645161
293  aberdeen         2021-05-31  69.794118
294  aberdeen         2021-08-31  64.969697

[295 rows x 3 columns]
```

[38]:
```
#There does not seem to be a pattern for Aberdeen.
aberdeen.plot(x='Date of Assessment', y='Mean')
```

[38]: <AxesSubplot:xlabel='Date of Assessment'>




### 1.5 4 Rank Towns by potential environmental impact rating 'Potential Environmental Impact Rating'

[39]:
```
town_pot_env = dataset.groupby(['POST_TOWN'])['Potential Environmental Impact␣
↪Rating'].mean().reset_index(name='Mean')
town_pot_env = town_pot_env.sort_values(by='Mean', ascending=False).
↪reset_index(drop=True)
town_pot_env.head()
```

[39]:

| | POST_TOWN | Mean |
|---|---|---|
| 0 | west plean | 126.0 |
| 1 | meigle | 122.0 |
| 2 | comrie | 121.0 |
| 3 | gartocharn | 117.0 |
| 4 | sanday | 117.0 |

### 1.6 5 Rank Towns by Current Emissions (T.CO2/yr)

[40]:
```
town_curr_em = dataset.groupby(['POST_TOWN'])['Current Emissions (T.CO2/yr)'].
↪mean().reset_index(name='Mean')
town_curr_em = town_pot_env.sort_values(by='Mean', ascending=False).
↪reset_index(drop=True)
town_curr_em.head()
```

```
[40]:      POST_TOWN   Mean
      0  west plean  126.0
      1      meigle  122.0
      2      comrie  121.0
      3      sanday  117.0
      4  gartocharn  117.0
```

```
[41]: dataset.columns
```

```
[41]: Index(['Property_UPRN', 'Postcode', 'POST_TOWN', 'Date of Assessment',
       'Primary Energy Indicator (kWh/m²/year)', 'Total floor area (m²)',
       'Current energy efficiency rating',
       'Current energy efficiency rating band',
       'Potential Energy Efficiency Rating',
       'Potential energy efficiency rating band',
       'Current Environmental Impact Rating',
       'Current Environmental Impact Rating Band',
       'Potential Environmental Impact Rating',
       'Potential Environmental Impact Rating Band',
       'CO2 Emissions Current Per Floor Area (kg.CO2/m²/yr)',
       'WALL_DESCRIPTION', 'WALL_ENERGY_EFF', 'ROOF_DESCRIPTION',
       'ROOF_ENERGY_EFF', 'FLOOR_DESCRIPTION', 'FLOOR_ENERGY_EFF',
       'FLOOR_ENV_EFF', 'WINDOWS_DESCRIPTION', 'WINDOWS_ENERGY_EFF',
       'WINDOWS_ENV_EFF', 'MAINHEAT_DESCRIPTION', 'MAINHEAT_ENERGY_EFF',
       'MAINHEAT_ENV_EFF', 'MAINHEATCONT_DESCRIPTION', 'MAINHEATC_ENERGY_EFF',
       'MAINHEATC_ENV_EFF', 'HOT_WATER_ENERGY_EFF', 'HOT_WATER_ENV_EFF',
       'LIGHTING_DESCRIPTION', 'LIGHTING_ENERGY_EFF', 'LIGHTING_ENV_EFF',
       'Current Emissions (T.CO2/yr)',
       'Potential Reduction in Emissions (T.CO2/yr)',
       'Total current energy costs over 3 years (£)',
       'Current heating costs over 3 years (£)',
       'Potential heating costs over 3 years (£)',
       'Current hot water costs over 3 years (£)',
       'Potential hot water costs over 3 years (£)',
       'Current lighting costs over 3 years (£)',
       'Potential lighting costs over 3 years (£)',
       'Part 1 Construction Age Band', 'Built Form', 'Property Type'],
      dtype='object')
```

## 1.7 6 Rank Towns by Potential Reduction in Emissions (T.CO2/yr)

```
[42]: town_pot_red_em = dataset.groupby(['POST_TOWN'])['Potential Reduction in␣
       ↪Emissions (T.CO2/yr)'].mean().reset_index(name='Mean')
      town_pot_red_em = town_pot_env.sort_values(by='Mean', ascending=False).
       ↪reset_index(drop=True)
      town_pot_red_em.head()
```

[42]:
```
     POST_TOWN   Mean
0  west plean  126.0
1      meigle  122.0
2      comrie  121.0
3      sanday  117.0
4  gartocharn  117.0
```

### 1.8 7 Rank Towns by potential savings in heating costs (£) over three years

[43]:
```python
dataset['heat_savings'] = dataset['Current heating costs over 3 years (£)'] -␣
↪dataset['Potential heating costs over 3 years (£)']
```

[44]:
```python
town_save_heating = dataset.groupby(['POST_TOWN'])['heat_savings'].mean().
↪reset_index(name='Mean')
town_save_heating = town_save_heating.sort_values(by='Mean', ascending=False).
↪reset_index(drop=True)
town_save_heating.head()
```

[44]:
```
             POST_TOWN     Mean
0               bonawe  11085.0
1  east dunbartonshire  11016.0
2              morvern   7098.0
3              corsock   6639.0
4             findhorn   5922.0
```

### 1.9 8 Rank Towns by potential savings in hot water costs (£) over three years

[45]:
```python
dataset['hot_water_save'] = dataset['Current hot water costs over 3 years (£)']␣
↪- dataset['Potential hot water costs over 3 years (£)']
town_hot_water_save = dataset.groupby(['POST_TOWN'])['hot_water_save'].mean().
↪reset_index(name='Mean')
town_hot_water_save = town_hot_water_save.sort_values(by='Mean',␣
↪ascending=False).reset_index(drop=True)
town_hot_water_save.head()
```

[45]:
```
             POST_TOWN    Mean
0           kincardine  2283.0
1  east dunbartonshire  1653.0
2           by maybole  1080.0
3              corsock  1056.0
4             burghead   978.0
```

## 1.10 9 Rank the top 5 wall descriptions (wall materials) by CO2 emissions current per floor area and wall energy efficiency

## 1.11 (create a single rating combining CO2 emissions and wall energy efficiency)

[46]:
```
dataset['WALL_ENERGY_EFF'].value_counts()
```

[46]:
```
Good                                33397
Very Good                           26451
Average                             19903
Poor                                18015
Poor | Poor                          9256
                                    …
Good | Very Poor | Average               1
Very Good | Poor | Average               1
Poor | Average | Poor | Good             1
Very Poor | Average | Very Good          1
Very Good | Good | Average               1
Name: WALL_ENERGY_EFF, Length: 261, dtype: int64
```

[47]:
```
#working this off in Excel
#The idea is to average the readings separate by pipes with a lookup table.
wall_energy_eff = dataset['WALL_ENERGY_EFF'].reset_index(drop=True)
wall_energy_eff.to_csv('wall_energy_eff')
```

[48]:
```
wall_ee = pd.read_csv('wall_energy_eff.csv')
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3444: DtypeWarning: Columns (4,13)
have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

[49]:
```
wall_ee.columns
```

[49]:
```
Index(['index', 'rating a ', ' rating b ', ' rating c ', ' rating d ',
       'rating a_num', 'rating b_num', 'rating c_num', 'rating d_num',
       'valid_cells', 'Value', 'Average_rating', 'Unnamed: 12', 'Rating',
       'Score'],
      dtype='object')
```

[50]:
```
dataset['AGG_RATING'] = wall_ee['Average_rating']
```

This is the first time, for subsequent runs, should replace this EE_PRODUCT with WALL_EE_PRODUCT as there is a roof version later on

Create new rating - wall emissions-efficiency product (EE_PRODUCT)
The lower the emissions, the better.
The lower the rating, the better.
Very Good = 5
Good = 4

Average = 3
Poor = 4
Very Poor = 5

[51]:
```
#creating the emissions-energy efficency (EE) product
dataset['EE_PRODUCT'] = dataset['CO2 Emissions Current Per Floor Area (kg.CO2/
↪m²/yr)'] * dataset['AGG_RATING']
```

[52]:
```
dataset['WALL_DESCRIPTION'].value_counts()
```

[52]:
```
Cavity wall, filled cavity
21917
Timber frame, as built, insulated (assumed)
18387
Cavity wall, as built, no insulation (assumed)
10100
Sandstone or limestone, as built, no insulation (assumed) | Solid brick, as
built, no insulation (assumed)
7881
Cavity wall, as built, insulated (assumed)
7836
…
Granite or whinstone, as built, insulated (assumed) | Sandstone or limestone, as
built, no insulation (assumed)
1
Solid brick, as built, no insulation (assumed) | Sandstone or limestone, with
internal insulation
1
Granite or whinstone, as built, no insulation (assumed) | Solid brick, as built,
insulated (assumed) | System built, as built, partial insulation (assumed)
1
Cavity wall, filled cavity | Granite or whinstone, as built, insulated (assumed)
1
Cavity wall, with internal insulation | Granite or whinstone, with internal
insulation | Timber frame, as built, insulated (assumed)
1
Name: WALL_DESCRIPTION, Length: 1519, dtype: int64
```

[53]:
```
#Outcome - the relationship between wall descriptions and the emissions-energy␣
↪efficency (EE) product
wall_desc_ee = dataset.groupby(['WALL_DESCRIPTION'])['EE_PRODUCT'].mean().
↪reset_index(name='Mean')
wall_desc_ee = wall_desc_ee.sort_values(by='Mean', ascending=True).
↪reset_index(drop=True)
wall_desc_ee
```

[53]:
```
                                     WALL_DESCRIPTION        Mean
0                Average thermal transmittance 0.09 W/m²K  -12.500000
1               Average thermal transmittance 0.14 W/mÂ²K   15.000000
2     Cavity wall, filled cavity | Granite or whinst…   22.000000
3     Granite or whinstone, as built, insulated (ass…   27.000000
4     Cavity wall, with internal insulation | Timber…   31.500000
…                                                 …         …
1514  Granite or whinstone, as built, no insulation …  744.000000
1515  Granite or whinstone, as built, no insulation …  806.000000
1516  Cavity wall, as built, partial insulation (ass…  832.000000
1517  Timber frame, as built, no insulation (assumed…  895.500000
1518  Granite or whinstone, as built, no insulation …  957.666667

[1519 rows x 2 columns]
```

### 1.12 10 Rank the top 5 roof descriptions by CO2 emissions current per floor area and wall energy efficiency

### 1.13 (create a single rating combining CO2 emissions and wall energy efficiency)

[54]:
```python
dataset['ROOF_ENERGY_EFF'].value_counts()
```

[54]:
```
Good                                    40333
Very Good                               26581
N/A                                     11929
Good                                    11390
Very Poor                                7909
                                        …
N/A | Poor | Good                           1
N/A | Very Good | Very Poor                 1
Very Poor | Average | Very Good             1
Average | Very Poor | Average               1
Average | Poor | Very Good                  1
Name: ROOF_ENERGY_EFF, Length: 294, dtype: int64
```

[55]:
```python
#export to work off separately in Excel
#vlookup to convert the strings into values, and then to create an average value
roof_energy_eff = dataset['ROOF_ENERGY_EFF'].to_csv('roof_ee.csv')
```

[56]:
```python
roof_ee = pd.read_csv('roof_ee_new.csv')
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3444: DtypeWarning: Columns (17) have
mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

[57]:
```python
roof_ee.head()
```

[57]:
```
   Index  rating_a rating_b   rating_c  rating_a_num  rating_b_num  \
0      0      Poor      NaN        NaN           4.0           NaN
1      1   Average     Good   VeryPoor           3.0           3.0
2      2                NaN        NaN           NaN           NaN
3      3      Good      NaN        NaN           3.0           NaN
4      4      Good      NaN        NaN           3.0           NaN

   rating_c_num  valid_cells  value average_rating  Unnamed: 10  Unnamed: 11  \
0           NaN            1      4              4          NaN          NaN
1           5.0            3     11    3.666666667          NaN          NaN
2           NaN            0      0        #DIV/0!          NaN          NaN
3           NaN            1      3              3          NaN          NaN
4           NaN            1      3              3          NaN          NaN

   Unnamed: 12  Unnamed: 13  Unnamed: 14  Unnamed: 15  Unnamed: 16     Value  \
0          NaN          NaN          NaN          NaN          NaN  VeryGood
1          NaN          NaN          NaN          NaN          NaN      Good
2          NaN          NaN          NaN          NaN          NaN   Average
3          NaN          NaN          NaN          NaN          NaN      Poor
4          NaN          NaN          NaN          NaN          NaN  VeryPoor

   Score
0    1.0
1    2.0
2    3.0
3    4.0
4    5.0
```

[58]:
```python
roof_ee['average_rating'].replace('#DIV/0!',value=-1)
```

[58]:
```
0                   4
1         3.666666667
2                  -1
3                   3
4                   3
             …
185034              4
185035              3
185036              3
185037             -1
185038             -1
Name: average_rating, Length: 185039, dtype: object
```

[59]:
```python
dataset['ROOF_RATING'] = roof_ee['average_rating']
```

[60]:
```python
dataset['ROOF_RATING'] = dataset['ROOF_RATING'].replace('#DIV/0!',value=-1)
```

[61]:
```
dataset['ROOF_RATING'] = dataset['ROOF_RATING'].astype(float)
```

[62]:
```
dataset['ROOF_EE_PRODUCT'] = dataset['CO2 Emissions Current Per Floor Area (kg.
↪CO2/m²/yr)'] * dataset['ROOF_RATING']
```

[63]:
```
roof_desc = dataset.groupby(['ROOF_DESCRIPTION'])['ROOF_EE_PRODUCT'].mean().
↪reset_index(name='Mean')
roof_desc = roof_desc.sort_values(by='Mean', ascending=True).
↪reset_index(drop=True)
roof_desc

#ignore the initial negative numbers, as they really mean negative numbers
```

[63]:
```
                                          ROOF_DESCRIPTION         Mean
0     (another dwelling above) | Pitched, 100 mm lof…   -61.892857
1     (another dwelling above) | Pitched, 100 mm lof…   -51.000000
2                             (another dwelling above)    -42.876719
3                              (another dwelling above)   -42.743910
4     (another dwelling above) | Pitched, 75 mm loft…   -37.000000
…                                                  …          …
1979  Pitched, 25 mm loft insulation | Pitched, 50 m…   853.666667
1980  Pitched, limited insulation (assumed) | Pitche…   868.000000
1981  Pitched, 250 mm loft insulation | Pitched, ins…   883.666667
1982  Roof room(s), ceiling insulated | Pitched, 100…   905.000000
1983  Pitched, 50 mm loft insulation | Pitched, no i…  1190.000000

[1984 rows x 2 columns]
```

[64]:
```
roof_desc[9:15]
```

[64]:
```
                                       ROOF_DESCRIPTION       Mean
9   (another dwelling above) | Pitched, insulated …   8.200000
10  Flat, no insulation (assumed) | Pitched, 300 m…  12.000000
11  Pitched, 400 mm loft insulation | Roof room(s)…  14.000000
12  Flat, insulated (assumed) | Pitched, 400 mm lo…  23.333333
13  Pitched, 200 mm loft insulation | Pitched, 400…  23.333333
14  (another dwelling above) | Flat, insulated (as…  27.745098
```

# 2 Part 2: Algorithm Challenges

## 2.1 Algorithm Challenge 1: Build and algorithm to find correlations between CO2 emissions current per floor area vs wall description and wall energy efficiency

[65]:
```
#The same groupby dataframe from Part 1 Challenge 9 because the same indicators␣
↪are used.
wall_desc = dataset['WALL_DESCRIPTION']
```

[66]:
```
#implementation from https://www.toptal.com/python/topic-modeling-python
#using LDA to create topic models from the wall descriptions

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.decomposition import LatentDirichletAllocation as LDA
from nltk.corpus import stopwords
```

[67]:
```
#approach from https://www.toptal.com/python/topic-modeling-python

corpus = wall_desc_ee['WALL_DESCRIPTION']
```

[68]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\eddie\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[68]:
```
True
```

[69]:
```
count_vect = CountVectorizer(stop_words=stopwords.words('english'),␣
↪lowercase=True)
x_counts = count_vect.fit_transform(corpus)
x_counts.todense()
```

[69]:
```
matrix([[0, 0, 0, …, 1, 0, 0],
        [0, 0, 0, …, 1, 0, 0],
        [0, 0, 0, …, 0, 1, 1],
        …,
        [0, 0, 0, …, 0, 1, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 1]], dtype=int64)
```

[70]:
```
count_vect.get_feature_names()
```

[70]:
```
['00',
 '06',
```

```
'07',
'09',
'10',
'11',
'12',
'13',
'14',
'15',
'16',
'17',
'18',
'19',
'20',
'21',
'22',
'23',
'24',
'25',
'26',
'27',
'28',
'29',
'30',
'31',
'33',
'35',
'36',
'38',
'40',
'41',
'42',
'43',
'45',
'46',
'47',
'48',
'51',
'52',
'53',
'55',
'56',
'57',
'58',
'60',
'62',
'63',
'64',
```

```
'65',
'66',
'67',
'68',
'70',
'71',
'72',
'73',
'74',
'76',
'77',
'78',
'79',
'80',
'81',
'83',
'84',
'92',
'99',
'additional',
'assumed',
'average',
'brick',
'built',
'cavity',
'cob',
'external',
'filled',
'frame',
'granite',
'home',
'insulated',
'insulation',
'internal',
'limestone',
'm²k',
'mâ²k',
'park',
'partial',
'sandstone',
'solid',
'system',
'thermal',
'timber',
'transmittance',
'wall',
'whinstone']
```

[71]:
```
tfidf_transformer = TfidfTransformer()
x_tfidf = tfidf_transformer.fit_transform(x_counts)
```

[72]:
```
dimension = 10
lda = LDA(n_components = dimension)
lda_array = lda.fit_transform(x_tfidf)
lda_array
```

[72]:
```
array([[0.03223826, 0.03223826, 0.03223826, …, 0.03223826, 0.03223826,
        0.03225258],
       [0.03174458, 0.03174458, 0.03174458, …, 0.03174458, 0.03174458,
        0.71429315],
       [0.02348672, 0.02350661, 0.02348661, …, 0.42116729, 0.39092268,
        0.02348093],
       …,
       [0.45022049, 0.35132919, 0.02480839, …, 0.02481099, 0.02480709,
        0.0248011 ],
       [0.42360646, 0.35654825, 0.02748175, …, 0.02748015, 0.02748168,
        0.02747578],
       [0.0269932 , 0.02701616, 0.02700624, …, 0.02699273, 0.7570274 ,
        0.02698739]])
```

[73]:
```
components = [lda.components_[i] for i in range(len(lda.components_))]
features = count_vect.get_feature_names()
important_words = [sorted(features, key = lambda x: components[j][features.
↪index(x)], reverse = True)[:3] for j in range(len(components))]
important_words
```

[73]:
```
[['timber', 'frame', 'built'],
 ['brick', 'solid', 'insulation'],
 ['system', 'built', 'insulation'],
 ['additional', 'timber', 'frame'],
 ['transmittance', 'average', 'thermal'],
 ['built', 'assumed', 'whinstone'],
 ['limestone', 'sandstone', 'built'],
 ['cavity', 'wall', 'filled'],
 ['granite', 'whinstone', 'built'],
 ['average', 'thermal', 'transmittance']]
```

[74]:
```
lda_array.shape
```

[74]:
```
(1519, 10)
```

[75]:
```
important_words[6]
```

[75]:
```
['limestone', 'sandstone', 'built']
```

[76]:
```python
list_max_wall_ee = []

for i in range(len(lda_array)):
    list_array = list(lda_array[i])
    max_num = list_array.index(max(list_array))
    list_max_wall_ee.append(max_num)

print(len(list_max_wall_ee))
```

```
1519
```

[77]:
```python
wall_desc_ee['description_number'] = list_max_wall_ee
```

[78]:
```python
wall_desc_ee.head(10)
```

[78]:
```
                                    WALL_DESCRIPTION  Mean  description_number
0            Average thermal transmittance 0.09 W/m²K -12.5                   4
1           Average thermal transmittance 0.14 W/mÂ²K  15.0                   9
2  Cavity wall, filled cavity | Granite or whinst…  22.0                   7
3  Granite or whinstone, as built, insulated (ass…  27.0                   8
4  Cavity wall, with internal insulation | Timber…  31.5                   3
5            Average thermal transmittance 0.18 W/m?K  35.0                   9
6  Granite or whinstone, as built, partial insula…  36.0                   1
7            Average thermal transmittance 0.13 W/m?K  40.0                   9
8            Average thermal transmittance 0.15 W/m?K  40.0                   9
9            Average thermal transmittance 0.43 W/m²K  43.5                   9
```

[79]:
```python
wall_desc_ee.dtypes
```

[79]:
```
WALL_DESCRIPTION       object
Mean                  float64
description_number      int64
dtype: object
```

[80]:
```python
wall_ee_short_des = []

for i in range(len(wall_desc_ee)):
    number = wall_desc_ee['description_number'][i]
    description = important_words[number]
    wall_ee_short_des.append(description)

print(len(wall_ee_short_des))
```

```
1519
```

[81]:
```python
wall_desc_ee["short_description"] = wall_ee_short_des
```

[82]:
```python
wall_desc_ee.head(10)
```

```
[82]:                                WALL_DESCRIPTION  Mean  \
      0          Average thermal transmittance 0.09 W/m²K -12.5
      1         Average thermal transmittance 0.14 W/mÂ²K  15.0
      2  Cavity wall, filled cavity | Granite or whinst…  22.0
      3  Granite or whinstone, as built, insulated (ass…  27.0
      4  Cavity wall, with internal insulation | Timber…  31.5
      5          Average thermal transmittance 0.18 W/m?K  35.0
      6  Granite or whinstone, as built, partial insula…  36.0
      7          Average thermal transmittance 0.13 W/m?K  40.0
      8          Average thermal transmittance 0.15 W/m?K  40.0
      9          Average thermal transmittance 0.43 W/m²K  43.5

         description_number                   short_description
      0                   4  [transmittance, average, thermal]
      1                   9  [average, thermal, transmittance]
      2                   7              [cavity, wall, filled]
      3                   8         [granite, whinstone, built]
      4                   3         [additional, timber, frame]
      5                   9  [average, thermal, transmittance]
      6                   1          [brick, solid, insulation]
      7                   9  [average, thermal, transmittance]
      8                   9  [average, thermal, transmittance]
      9                   9  [average, thermal, transmittance]
```

```
[83]: description_corr = wall_desc_ee.groupby(['description_number'])['Mean'].mean().
       ↪reset_index(name='agg_mean')
      description_corr.sort_values(by='agg_mean')
```

```
[83]:    description_number    agg_mean
      4                   4   85.527030
      9                   9   89.122080
      7                   7  171.748836
      3                   3  179.854958
      1                   1  191.418012
      2                   2  205.477818
      6                   6  215.009181
      0                   0  232.060242
      5                   5  241.089506
      8                   8  255.058597
```

```
[84]: agg_wall_short_desc = []

      for i in range(len(description_corr)):
          number = description_corr['description_number'][i]
          description = important_words[number]
          agg_wall_short_desc.append(description)
```

```
print(len(agg_wall_short_desc))
```

```
10
```

[85]:
```
description_corr['short_desc'] = agg_wall_short_desc
```

[86]:
```
description_corr
```

[86]:
```
   description_number    agg_mean                        short_desc
0                   0  232.060242            [timber, frame, built]
1                   1  191.418012        [brick, solid, insulation]
2                   2  205.477818       [system, built, insulation]
3                   3  179.854958       [additional, timber, frame]
4                   4   85.527030  [transmittance, average, thermal]
5                   5  241.089506       [built, assumed, whinstone]
6                   6  215.009181     [limestone, sandstone, built]
7                   7  171.748836            [cavity, wall, filled]
8                   8  255.058597       [granite, whinstone, built]
9                   9   89.122080  [average, thermal, transmittance]
```

[87]:
```
description_corr = description_corr.sort_values(by='agg_mean', ascending=True).
 ↪reset_index(drop=True)
description_corr
```

[87]:
```
   description_number    agg_mean                        short_desc
0                   4   85.527030  [transmittance, average, thermal]
1                   9   89.122080  [average, thermal, transmittance]
2                   7  171.748836            [cavity, wall, filled]
3                   3  179.854958       [additional, timber, frame]
4                   1  191.418012        [brick, solid, insulation]
5                   2  205.477818       [system, built, insulation]
6                   6  215.009181     [limestone, sandstone, built]
7                   0  232.060242            [timber, frame, built]
8                   5  241.089506       [built, assumed, whinstone]
9                   8  255.058597       [granite, whinstone, built]
```

### 2.2 Algorithm Challenge 2: Build and algorithm to find correlations between CO2 emissions current per floor area vs roof description and roof energy efficiency

[88]:
```
#do the same topic model for roof descriptions
#Starting on Part 2 Challenge 2. The indicator from Part 1 #10 is used as they␣
 ↪are the same.
roof_corpus = roof_desc['ROOF_DESCRIPTION']
```

[89]:
```
roof_count_vect = CountVectorizer(stop_words=stopwords.words('english'),␣
 ↪lowercase=True)
roof_counts = roof_count_vect.fit_transform(roof_corpus)
```

```
roof_counts.todense()
```

[89]:
```
matrix([[0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        …,
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0]], dtype=int64)
```

[90]:
```
roof_count_vect.get_feature_names()
```

[90]:
```
['05',
 '06',
 '07',
 '08',
 '09',
 '10',
 '100',
 '11',
 '12',
 '13',
 '14',
 '15',
 '150',
 '16',
 '17',
 '18',
 '19',
 '20',
 '200',
 '21',
 '22',
 '23',
 '24',
 '25',
 '250',
 '270',
 '29',
 '300',
 '31',
 '32',
 '35',
 '350',
 '38',
 '40',
 '400',
```

```
 '50',
 '57',
 '75',
 '81',
 'additional',
 'another',
 'assumed',
 'average',
 'ceiling',
 'code',
 'dwelling',
 'flat',
 'input',
 'insulated',
 'insulation',
 'invalid',
 'limited',
 'loft',
 'mm',
 'm²k',
 'mâ²k',
 'pitched',
 'premises',
 'rafters',
 'roof',
 'room',
 'thatched',
 'thermal',
 'transmittance']
```

[91]:
```
tfidf_transformer = TfidfTransformer()
roof_tfidf = tfidf_transformer.fit_transform(roof_counts)
```

[92]:
```
dimension = 10
roof_lda = LDA(n_components = dimension)
roof_lda_array = roof_lda.fit_transform(roof_tfidf)
roof_lda_array
```

[92]:
```
array([[0.03008411, 0.03008561, 0.03007957, …, 0.03007827, 0.49806048,
        0.03008409],
       [0.03008411, 0.03008561, 0.03007957, …, 0.03007827, 0.49806048,
        0.03008409],
       [0.04142136, 0.04142136, 0.04142136, …, 0.04142136, 0.62720779,
        0.04142136],
       …,
       [0.02784578, 0.02784552, 0.27840889, …, 0.02783814, 0.0278427 ,
        0.02784572],
```

```
[0.02506655, 0.02506658, 0.31475361, …, 0.02505973, 0.02506373,
 0.02506699],
[0.02793866, 0.74856102, 0.02793915, …, 0.02793059, 0.02793528,
 0.02793882]])
```

[93]:
```python
print(len(roof_lda_array))
```

```
1984
```

[94]:
```python
roof_components = [roof_lda.components_[i] for i in range(len(roof_lda.
 ↪components_))]
roof_features = roof_count_vect.get_feature_names()
```

[95]:
```python
roof_features
```

[95]:
```
['05',
 '06',
 '07',
 '08',
 '09',
 '10',
 '100',
 '11',
 '12',
 '13',
 '14',
 '15',
 '150',
 '16',
 '17',
 '18',
 '19',
 '20',
 '200',
 '21',
 '22',
 '23',
 '24',
 '25',
 '250',
 '270',
 '29',
 '300',
 '31',
 '32',
 '35',
 '350',
 '38',
```

```
 '40',
 '400',
 '50',
 '57',
 '75',
 '81',
 'additional',
 'another',
 'assumed',
 'average',
 'ceiling',
 'code',
 'dwelling',
 'flat',
 'input',
 'insulated',
 'insulation',
 'invalid',
 'limited',
 'loft',
 'mm',
 'm²k',
 'mâ²k',
 'pitched',
 'premises',
 'rafters',
 'roof',
 'room',
 'thatched',
 'thermal',
 'transmittance']
```

[96]:
```
len(roof_features)
```

[96]: 64

[97]:
```
len(roof_components)
```

[97]: 10

[98]:
```
roof_important_words = [sorted(roof_features, key = lambda x:␣
↪roof_components[j][roof_features.index(x)], reverse = True)[:3] for j in␣
↪range(len(roof_components))]
roof_important_words
```

[98]:
```
[['200', 'insulation', 'pitched'],
 ['insulation', 'pitched', 'loft'],
```

```
 ['insulated', 'ceiling', 'room'],
 ['300', 'insulation', 'pitched'],
 ['assumed', 'insulation', 'limited'],
 ['270', 'insulation', 'pitched'],
 ['insulation', 'pitched', 'mm'],
 ['average', 'thermal', 'transmittance'],
 ['another', 'dwelling', '350'],
 ['150', 'insulation', 'pitched']]
```

[99]:
```
list_max_roof_ee = []

for i in range(len(roof_lda_array)):
    list_array = list(roof_lda_array[i])
    max_num = list_array.index(max(list_array))
    list_max_roof_ee.append(max_num)

print(len(list_max_roof_ee))
```

```
1984
```

[100]:
```
roof_desc['desc_num'] = list_max_roof_ee
```

[101]:
```
roof_description_corr = roof_desc.groupby(['desc_num'])['Mean'].mean().
 ↪reset_index(name='agg_mean')
roof_description_corr = roof_description_corr.sort_values(by='agg_mean')
roof_description_corr
```

[101]:

| | desc_num | agg_mean |
|---|---|---|
| 7 | 7 | 73.692240 |
| 8 | 8 | 209.731351 |
| 5 | 5 | 215.616475 |
| 2 | 2 | 219.568682 |
| 9 | 9 | 224.539906 |
| 0 | 0 | 226.094061 |
| 3 | 3 | 229.226621 |
| 1 | 1 | 239.012800 |
| 6 | 6 | 240.832711 |
| 4 | 4 | 286.202318 |

[102]:
```
agg_roof_short_desc = []

for i in range(len(roof_description_corr)):
    number = roof_description_corr['desc_num'][i]
    description = roof_important_words[number]
    agg_roof_short_desc.append(description)

print(len(agg_roof_short_desc))
```

```
10
```

[103]:
```
roof_description_corr['short_desc'] = agg_roof_short_desc
```

[104]:
```
#Outcome of Algorithm Challenge 2: There is some relationship between the␣
↪descriptions and the roof energy efficiency and emissions (the lower the␣
↪figure, the better the energy performance)
roof_description_corr
```

[104]:
```
   desc_num    agg_mean                         short_desc
7         7   73.692240        [200, insulation, pitched]
8         8  209.731351       [insulation, pitched, loft]
5         5  215.616475        [insulated, ceiling, room]
2         2  219.568682        [300, insulation, pitched]
9         9  224.539906    [assumed, insulation, limited]
0         0  226.094061        [270, insulation, pitched]
3         3  229.226621         [insulation, pitched, mm]
1         1  239.012800  [average, thermal, transmittance]
6         6  240.832711          [another, dwelling, 350]
4         4  286.202318        [150, insulation, pitched]
```

### 2.3 Algorithm Challenge 3: Build and algorithm to find correlations between construction age band vs current energy efficiency and current emissions (T.CO2/yr)

The approach to this is to create a linear regression model using the age of the building and to find out the relationship between current energy efficiency and current emissions.

[105]:
```
dataset.columns
```

[105]:
```
Index(['Property_UPRN', 'Postcode', 'POST_TOWN', 'Date of Assessment',
       'Primary Energy Indicator (kWh/m²/year)', 'Total floor area (m²)',
       'Current energy efficiency rating',
       'Current energy efficiency rating band',
       'Potential Energy Efficiency Rating',
       'Potential energy efficiency rating band',
       'Current Environmental Impact Rating',
       'Current Environmental Impact Rating Band',
       'Potential Environmental Impact Rating',
       'Potential Environmental Impact Rating Band',
       'CO2 Emissions Current Per Floor Area (kg.CO2/m²/yr)',
       'WALL_DESCRIPTION', 'WALL_ENERGY_EFF', 'ROOF_DESCRIPTION',
       'ROOF_ENERGY_EFF', 'FLOOR_DESCRIPTION', 'FLOOR_ENERGY_EFF',
       'FLOOR_ENV_EFF', 'WINDOWS_DESCRIPTION', 'WINDOWS_ENERGY_EFF',
       'WINDOWS_ENV_EFF', 'MAINHEAT_DESCRIPTION', 'MAINHEAT_ENERGY_EFF',
       'MAINHEAT_ENV_EFF', 'MAINHEATCONT_DESCRIPTION', 'MAINHEATC_ENERGY_EFF',
       'MAINHEATC_ENV_EFF', 'HOT_WATER_ENERGY_EFF', 'HOT_WATER_ENV_EFF',
       'LIGHTING_DESCRIPTION', 'LIGHTING_ENERGY_EFF', 'LIGHTING_ENV_EFF',
```

```
        'Current Emissions (T.CO2/yr)',
        'Potential Reduction in Emissions (T.CO2/yr)',
        'Total current energy costs over 3 years (£)',
        'Current heating costs over 3 years (£)',
        'Potential heating costs over 3 years (£)',
        'Current hot water costs over 3 years (£)',
        'Potential hot water costs over 3 years (£)',
        'Current lighting costs over 3 years (£)',
        'Potential lighting costs over 3 years (£)',
        'Part 1 Construction Age Band', 'Built Form', 'Property Type',
        'heat_savings', 'hot_water_save', 'AGG_RATING', 'EE_PRODUCT',
        'ROOF_RATING', 'ROOF_EE_PRODUCT'],
       dtype='object')
```

[106]:
```
age_of_build = dataset.groupby(['Part 1 Construction Age Band'])['Current␣
↪energy efficiency rating','Current Emissions (T.CO2/yr)'].mean().
↪reset_index()
```

```
C:\Users\eddie\AppData\Local\Temp/ipykernel_11664/1828837144.py:1:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  age_of_build = dataset.groupby(['Part 1 Construction Age Band'])['Current
energy efficiency rating','Current Emissions (T.CO2/yr)'].mean().reset_index()
```

[107]:
```
age_of_build.head
```

[107]:
```
<bound method NDFrame.head of    Part 1 Construction Age Band  Current energy
efficiency rating  \
0                    1919-1929                          62.244228
1                    1930-1949                          64.988258
2                    1950-1964                          65.663387
3                    1965-1975                          66.230327
4                    1976-1983                          67.528634
5                    1984-1991                          69.085491
6                    1992-1998                          71.311040
7                    1999-2002                          72.708708
8                    2003-2007                          76.565927
9                 2008 onwards                          78.443133
10                 before 1919                          59.158740

    Current Emissions (T.CO2/yr)
0                       4.805747
1                       4.264428
2                       3.973697
3                       4.079704
4                       3.858721
5                       3.484475
6                       3.473204
```

```
7                         3.404447
8                         2.898558
9                         2.598642
10                        5.644251  >
```

[108]:
```
age_of_build['Year'] = [1924, 1940, 1957, 1970, 1979, 1987, 1995, 2000, 2005,␣
 ↪2015, 1919]
```

[109]:
```
age_of_build = age_of_build.sort_values(by='Year', ascending=True).
 ↪reset_index(drop=True)
age_of_build
```

[109]:
```
   Part 1 Construction Age Band  Current energy efficiency rating  \
0                   before 1919                          59.158740
1                     1919-1929                          62.244228
2                     1930-1949                          64.988258
3                     1950-1964                          65.663387
4                     1965-1975                          66.230327
5                     1976-1983                          67.528634
6                     1984-1991                          69.085491
7                     1992-1998                          71.311040
8                     1999-2002                          72.708708
9                     2003-2007                          76.565927
10                 2008 onwards                          78.443133

    Current Emissions (T.CO2/yr)  Year
0                         5.644251  1919
1                         4.805747  1924
2                         4.264428  1940
3                         3.973697  1957
4                         4.079704  1970
5                         3.858721  1979
6                         3.484475  1987
7                         3.473204  1995
8                         3.404447  2000
9                         2.898558  2005
10                        2.598642  2015
```

[110]:
```
age_of_build['est_building_age'] = 2022-age_of_build['Year']
```

[111]:
```
age_of_build
```

[111]:
```
   Part 1 Construction Age Band  Current energy efficiency rating  \
0                   before 1919                          59.158740
1                     1919-1929                          62.244228
2                     1930-1949                          64.988258
3                     1950-1964                          65.663387
```

```
4                      1965-1975                         66.230327
5                      1976-1983                         67.528634
6                      1984-1991                         69.085491
7                      1992-1998                         71.311040
8                      1999-2002                         72.708708
9                      2003-2007                         76.565927
10                  2008 onwards                         78.443133

    Current Emissions (T.CO2/yr)  Year  est_building_age
0                       5.644251  1919               103
1                       4.805747  1924                98
2                       4.264428  1940                82
3                       3.973697  1957                65
4                       4.079704  1970                52
5                       3.858721  1979                43
6                       3.484475  1987                35
7                       3.473204  1995                27
8                       3.404447  2000                22
9                       2.898558  2005                17
10                      2.598642  2015                 7
```

There is some looseness here, as buildings before 1919 are classified have a 1919 start date.

[112]:
```
year_dict = {0:0,"before 1919":1919, "1919-1929":1924, "1930-1949":
↪1940,"1950-1964":1957,"1965-1975":1970, "1976-1983":1979,"1984-1991":
↪1987,"1992-1998":1995,"1999-2002":2000,"2003-2007":2005, "2008 onwards":2015}
```

[113]:
```
dataset["Part 1 Construction Age Band"] = dataset["Part 1 Construction Age␣
↪Band"].fillna(0)
```

[114]:
```
all_building_year = []

for i in range(len(dataset)):
    year = year_dict[dataset['Part 1 Construction Age Band'][i]]
    all_building_year.append(year)

print(len(all_building_year))
```

```
185039
```

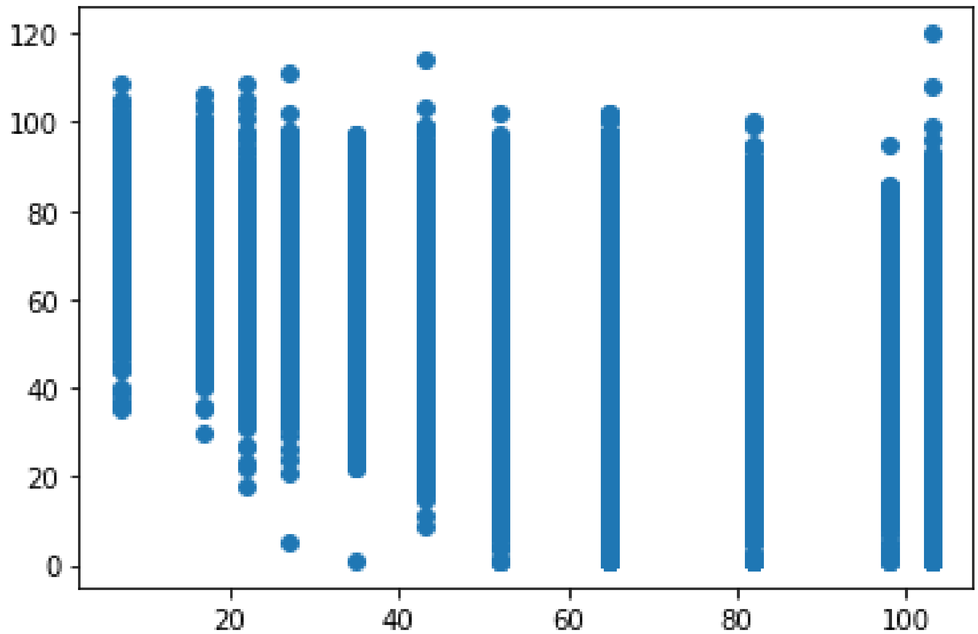
[115]:
```
dataset['est_build_year'] = all_building_year
```

[116]:
```
dataset['build_age'] = 2022-dataset['est_build_year']
```

[117]:
```
#for analysis, drop rows whhere build_age = 2022

analysis_build_age = dataset[dataset['build_age'] !=2022].reset_index(drop=True)
```

[118]:
```
plt.scatter(analysis_build_age['build_age'], analysis_build_age['Current energy␣
↪efficiency rating'])
plt.show()
```




[119]:
```
from sklearn.linear_model import LinearRegression
```

[120]:
```
len(analysis_build_age)
```

[120]: 155067

[121]:
```
0.8*len(analysis_build_age)
```

[121]: 124053.6

[122]:
```
0.2*len(analysis_build_age)
```

[122]: 31013.4

[123]:
```
analysis_build_age['build_age']
```

[123]:
```
0          82
1          98
2          52
3          22
4         103
         …
```

```
155062      7
155063     82
155064     35
155065     52
155066     52
Name: build_age, Length: 155067, dtype: int64
```

[124]:
```python
import numpy as np

X = np.array(analysis_build_age['build_age'])
y = analysis_build_age['Current energy efficiency rating']
```

[125]:
```python
X=X.reshape(-1,1)
```

[126]:
```python
lm = LinearRegression()
lm.fit(X, y)
```

[126]:
```
LinearRegression()
```

[127]:
```python
print(lm.coef_)
```

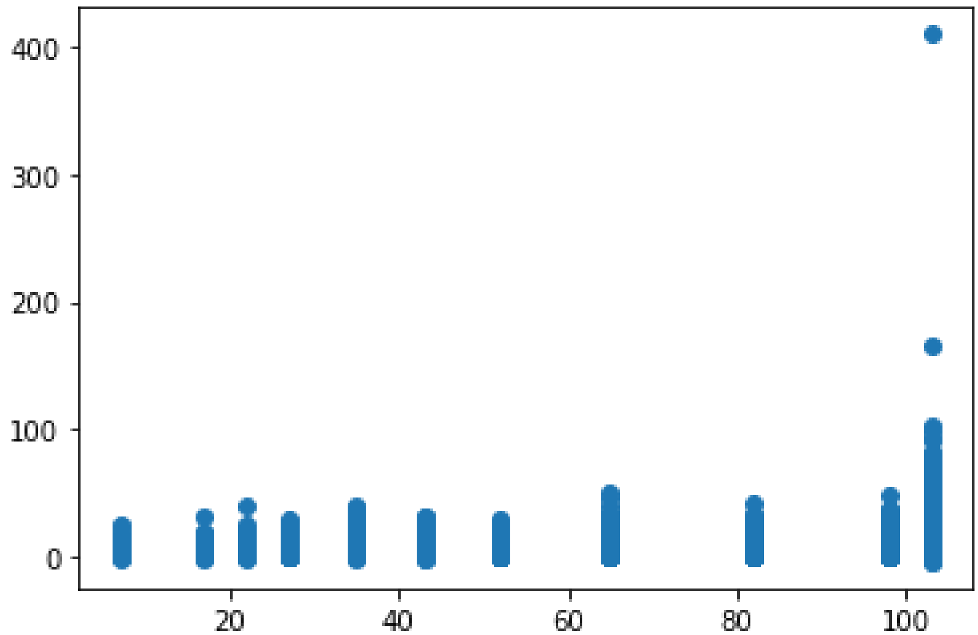
```
[-0.16721293]
```

[128]:
```python
print(lm.intercept_)
```

```
76.51658142966357
```

[129]:
```python
#Creating the regression model between building age and current emissions
z = analysis_build_age['Current Emissions (T.CO2/yr)']
```

[130]:
```python
plt.scatter(X, z)
plt.show()
```




[131]:
```
lm2 = LinearRegression()
lm2.fit(X, z)
```

[131]: LinearRegression()

[132]:
```
print(lm2.coef_)
print(lm2.intercept_)
```

```
[0.0280689]
2.4789921547361917
```

#### 2.3.1 For energy efficiency, the older the building, energy efficiency (higher the better) declines by 0.167 units. Also, the older the building, the higher the emissions (lower the better), by 0.028 units.

#### 2.3.2 As expected, the newer the building, the better the energy efficiency; the newer the building, emissions are lower.

### 2.4 Algorithm Challenge 4 Build an algorithm that takes as input the characteristics of a building (any field of the dataset) and outputs recommendations on the elements of the house to be modified to improve its energy performance - (15 points)

Code from the LDA process (same as 1 and 2) comes from: https://towardsdatascience.com/how-to-easily-cluster-textual-data-in-python-ab27040b07d8

The approach to this would be to instead, look at the best performers in terms of emissions and energy efficiency, and use the wall and roof descriptions to look at what good energy performers will look like.

The LDA approach will be used again to shorten the descriptions.

[133]:
```
dataset.dtypes
```

[133]:
```
Property_UPRN                                        float64
Postcode                                              object
POST_TOWN                                             object
Date of Assessment                                    object
Primary Energy Indicator (kWh/m²/year)               float64
Total floor area (m²)                                float64
Current energy efficiency rating                     float64
Current energy efficiency rating band                 object
Potential Energy Efficiency Rating                   float64
Potential energy efficiency rating band               object
Current Environmental Impact Rating                  float64
Current Environmental Impact Rating Band              object
Potential Environmental Impact Rating                float64
Potential Environmental Impact Rating Band            object
CO2 Emissions Current Per Floor Area (kg.CO2/m²/yr)    float64
WALL_DESCRIPTION                                      object
WALL_ENERGY_EFF                                       object
ROOF_DESCRIPTION                                      object
ROOF_ENERGY_EFF                                       object
FLOOR_DESCRIPTION                                     object
FLOOR_ENERGY_EFF                                      object
FLOOR_ENV_EFF                                         object
WINDOWS_DESCRIPTION                                   object
WINDOWS_ENERGY_EFF                                    object
WINDOWS_ENV_EFF                                       object
MAINHEAT_DESCRIPTION                                  object
MAINHEAT_ENERGY_EFF                                   object
MAINHEAT_ENV_EFF                                      object
MAINHEATCONT_DESCRIPTION                              object
MAINHEATC_ENERGY_EFF                                  object
MAINHEATC_ENV_EFF                                     object
HOT_WATER_ENERGY_EFF                                  object
HOT_WATER_ENV_EFF                                     object
LIGHTING_DESCRIPTION                                  object
LIGHTING_ENERGY_EFF                                   object
LIGHTING_ENV_EFF                                      object
Current Emissions (T.CO2/yr)                         float64
Potential Reduction in Emissions (T.CO2/yr)          float64
Total current energy costs over 3 years (£)          float64
Current heating costs over 3 years (£)               float64
Potential heating costs over 3 years (£)             float64
Current hot water costs over 3 years (£)             float64
```

```
Potential hot water costs over 3 years (£)             float64
Current lighting costs over 3 years (£)                float64
Potential lighting costs over 3 years (£)              float64
Part 1 Construction Age Band                            object
Built Form                                              object
Property Type                                           object
heat_savings                                           float64
hot_water_save                                         float64
AGG_RATING                                             float64
EE_PRODUCT                                             float64
ROOF_RATING                                            float64
ROOF_EE_PRODUCT                                        float64
est_build_year                                           int64
build_age                                                int64
dtype: object
```

[134]:
```
emissions = dataset.sort_values(by="Current Emissions (T.CO2/yr)").
 ↪reset_index(drop=True)
```

[135]:
```
emissions.head(20)
```

[135]:
```
    Property_UPRN  Postcode      POST_TOWN Date of Assessment  \
0    1.235876e+09  PA21 2DB  tighnabruaich         28/10/2021
1    1.235429e+09   KY2 6FN      kirkcaldy         11/06/2021
2    1.235052e+09  EH23 4PS     gorebridge         06/04/2021
3    1.235755e+09  EH23 4NN     gorebridge         08/04/2021
4    1.001273e+09  KA16 9LJ       newmilns         29/07/2021
5    1.235743e+09   G71 7FR        glasgow         23/03/2021
6    1.235888e+09  PH36 4HY      acharacle         18/11/2021
7    1.234627e+09  TD11 3NG           duns         09/11/2021
8    1.235783e+09   G83 8SD     gartocharn         26/05/2021
9    1.235014e+09   ML8 5NE        carluke         23/08/2021
10   1.235226e+09  FK10 3QD          alloa         24/02/2021
11   1.235862e+09  AB41 7PR          ellon         07/10/2021
12   1.234889e+09   KY4 9EJ    cowdenbeath         01/09/2021
13   1.235046e+09  AB45 2UL          banff         17/03/2021
14   1.001026e+09   DD8 2NR         forfar         30/05/2021
15   1.000561e+09  PH13 9HU    blairgowrie         10/04/2021
16   1.235776e+09  KW17 2AN         orkney         08/07/2021
17   1.234879e+09  KW15 1SS         orkney         07/05/2021
18   1.000797e+09   TD5 7PH          kelso         26/05/2021
19   1.235858e+09   FK7 0HX       stirling         28/09/2021

    Primary Energy Indicator (kWh/m²/year)  Total floor area (m²)  \
0                                   -858.0                  143.0
1                                   -145.0                  346.0
2                                    -60.0                  215.0
```

```
3                                          -60.0                  215.0
4                                          140.0                  199.0
5                                         -263.0                   75.0
6                                         -125.0                  146.0
7                                           88.0                  226.0
8                                          -61.0                  292.0
9                                          -53.0                  301.0
10                                         -99.0                  140.0
11                                         -49.0                  139.0
12                                        -129.0                   89.0
13                                         -98.0                  167.0
14                                         -48.0                  151.0
15                                         -69.0                  135.0
16                                          81.0                  120.0
17                                         -40.0                  217.0
18                                         158.0                  209.0
19                                         -46.0                  147.0

    Current energy efficiency rating Current energy efficiency rating band  \
0                              268.0                                     A
1                              141.0                                     A
2                              124.0                                     A
3                              124.0                                     A
4                               78.0                                     C
5                              143.0                                     A
6                              126.0                                     A
7                               90.0                                     B
8                              115.0                                     A
9                              116.0                                     A
10                             122.0                                     A
11                             113.0                                     A
12                             128.0                                     A
13                              87.0                                     B
14                             120.0                                     A
15                              75.0                                     C
16                              96.0                                     A
17                             113.0                                     A
18                              73.0                                     C
19                             113.0                                     A

    Potential Energy Efficiency Rating  \
0                                291.0
1                                141.0
2                                134.0
3                                134.0
4                                 87.0
5                                144.0
```

```
6                               127.0
7                               102.0
8                               118.0
9                               117.0
10                              139.0
11                              131.0
12                              130.0
13                              100.0
14                              137.0
15                               94.0
16                              115.0
17                              119.0
18                               80.0
19                              113.0

   Potential energy efficiency rating band  …     Built Form Property Type  \
0                                        A  …       Detached         House
1                                        A  …       Detached      Bungalow
2                                        A  …       Detached      Bungalow
3                                        A  …       Detached      Bungalow
4                                        B  …       Detached         House
5                                        A  …  Semi-Detached         House
6                                        A  …       Detached         House
7                                        A  …       Detached         House
8                                        A  …       Detached         House
9                                        A  …       Detached         House
10                                       A  …       Detached      Bungalow
11                                       A  …       Detached         House
12                                       A  …       Detached         House
13                                       A  …  Semi-Detached         House
14                                       A  …       Detached      Bungalow
15                                       A  …       Detached      Bungalow
16                                       A  …       Detached      Bungalow
17                                       A  …       Detached         House
18                                       C  …       Detached         House
19                                       A  …  Semi-Detached      Bungalow

    heat_savings hot_water_save  AGG_RATING EE_PRODUCT ROOF_RATING  \
0            0.0          159.0    5.000000     -795.0         4.0
1            0.0            0.0    5.000000     -125.0         4.0
2            0.0            0.0    5.000000     -135.0         4.0
3            0.0            0.0    5.000000     -135.0         4.0
4          990.0          300.0    3.000000      -54.0         4.5
5            0.0           78.0    5.000000     -219.5         4.0
6            0.0          267.0    5.000000     -105.0         4.0
7          -27.0          294.0    4.500000      -58.5         3.0
8            0.0            0.0    5.000000      -51.5         4.0
```

```
9          -3.0       285.0  5.000000   -45.5      4.0
10         -3.0       294.0  5.000000   -85.0      4.0
11          0.0       435.0  5.000000   -73.5      3.0
12          0.0        78.0  5.000000  -105.0      4.0
13          0.0         0.0  5.000000   -55.0      4.0
14        171.0       294.0  3.500000   -38.5      4.0
15          0.0       411.0  3.333333   -40.0      4.0
16         -6.0       222.0  5.000000   -60.0      4.0
17         -3.0       285.0  5.000000   -35.0      4.0
18        759.0       252.0  3.000000   -15.0      3.0
19          0.0         0.0  5.000000   -35.0      3.0

   ROOF_EE_PRODUCT est_build_year build_age
0           -636.0              0      2022
1           -100.0              0      2022
2           -108.0              0      2022
3           -108.0              0      2022
4            -81.0           1919       103
5           -175.6              0      2022
6            -84.0              0      2022
7            -39.0           1919       103
8            -41.2              0      2022
9            -36.4              0      2022
10           -68.0              0      2022
11           -44.1              0      2022
12           -84.0              0      2022
13           -44.0              0      2022
14           -44.0           1919       103
15           -48.0           1919       103
16           -48.0              0      2022
17           -28.0              0      2022
18           -15.0           1987        35
19           -21.0              0      2022

[20 rows x 56 columns]
```

[136]:
```python
#Looking at the smallest emitters in CO2

emissions_1000 = emissions[:1000]
```

[137]:
```python
wall_ee_1000 = emissions_1000.groupby(['WALL_DESCRIPTION'])['Current energy␣
↪efficiency rating'].mean().reset_index(name="EE_WALL")
wall_ee_1000
```

[137]:
```
                               WALL_DESCRIPTION     EE_WALL
0          Average thermal transmittance 0.09 W/m²K  105.000000
1          Average thermal transmittance 0.11 W/m²K   97.000000
```

```
2            Average thermal transmittance 0.12 W/m²K  100.000000
3            Average thermal transmittance 0.13 W/m²K   92.777778
4           Average thermal transmittance 0.13 W/m²K    86.000000
..                                               …         …
81       Timber frame, as built, insulated (assumed)    86.870968
82  Timber frame, as built, insulated (assumed) | …   66.000000
83  Timber frame, as built, insulated (assumed) | …  101.000000
84  Timber frame, as built, partial insulation (as…   72.000000
85           Timber frame, with additional insulation  121.500000

[86 rows x 2 columns]
```

[138]:

```
#applying LDA to the wall description

em_1000_wall_ee = wall_ee_1000['WALL_DESCRIPTION']
count_vect = CountVectorizer(stop_words=stopwords.words('english'),␣
↪lowercase=True)
em_1000_wall_ee_vec = count_vect.fit_transform(em_1000_wall_ee)
em_1000_wall_ee_vec.todense()
```

[138]:

```
matrix([[1, 0, 0, …, 1, 0, 0],
        [0, 1, 0, …, 1, 0, 0],
        [0, 0, 1, …, 1, 0, 0],
        …,
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0],
        [0, 0, 0, …, 0, 0, 0]], dtype=int64)
```

[139]:

```
count_vect.get_feature_names()
```

[139]:

```
['09',
 '11',
 '12',
 '13',
 '14',
 '15',
 '16',
 '17',
 '18',
 '19',
 '20',
 '21',
 '22',
 '23',
 '25',
 'additional',
 'assumed',
```

```
 'average',
 'brick',
 'built',
 'cavity',
 'external',
 'filled',
 'frame',
 'granite',
 'insulated',
 'insulation',
 'internal',
 'limestone',
 'm²k',
 'mâ²k',
 'partial',
 'sandstone',
 'solid',
 'system',
 'thermal',
 'timber',
 'transmittance',
 'wall',
 'whinstone']
```

[140]:
```
em_1000_wall_ee_tfidf = tfidf_transformer.fit_transform(em_1000_wall_ee_vec)
```

[141]:
```
dimension = 5
em_1000_wall_ee_lda = LDA(n_components = dimension)
em_1000_wall_ee_lda_array = em_1000_wall_ee_lda.
 ↪fit_transform(em_1000_wall_ee_tfidf)
```

[142]:
```
em_1000_wall_ee_components = [em_1000_wall_ee_lda.components_[i] for i in␣
 ↪range(len(em_1000_wall_ee_lda.components_))]
features = count_vect.get_feature_names()
em_1000_wall_ee_important_words = [sorted(features, key = lambda x:␣
 ↪em_1000_wall_ee_components[j][features.index(x)], reverse = True)[:3] for j␣
 ↪in range(len(em_1000_wall_ee_components))]
em_1000_wall_ee_important_words
```

[142]:
```
[['cavity', 'wall', 'filled'],
 ['timber', 'frame', 'insulated'],
 ['built', 'insulation', 'sandstone'],
 ['average', 'thermal', 'transmittance'],
 ['m²k', 'average', 'thermal']]
```

[143]:
```
em_1000_wall_list = []
```

```
for i in range(len(em_1000_wall_ee_lda_array)):
    list_array = list(em_1000_wall_ee_lda_array[i])
    max_num = list_array.index(max(list_array))
    em_1000_wall_list.append(max_num)

print(len(em_1000_wall_list))
```

86

[144]:
```
wall_ee_1000['desc_num'] = em_1000_wall_list
```

[145]:
```
agg_em1000_wall_ee_short_desc = []

for i in range(len(wall_ee_1000)):
    number = wall_ee_1000['desc_num'][i]
    description = em_1000_wall_ee_important_words[number]
    agg_em1000_wall_ee_short_desc.append(description)

print(len(agg_em1000_wall_ee_short_desc))
```

86

[146]:
```
wall_ee_1000['short_desc']= agg_em1000_wall_ee_short_desc
```

[147]:
```
wall_ee_1000
```

[147]:
```
                                    WALL_DESCRIPTION     EE_WALL  desc_num  \
0          Average thermal transmittance 0.09 W/m²K  105.000000         3
1          Average thermal transmittance 0.11 W/m²K   97.000000         3
2          Average thermal transmittance 0.12 W/m²K  100.000000         3
3          Average thermal transmittance 0.13 W/m²K   92.777778         3
4         Average thermal transmittance 0.13 W/m²K    86.000000         3
..                                               …           …       …
81      Timber frame, as built, insulated (assumed)    86.870968         1
82  Timber frame, as built, insulated (assumed) | …   66.000000         1
83  Timber frame, as built, insulated (assumed) | …  101.000000         1
84  Timber frame, as built, partial insulation (as…   72.000000         1
85         Timber frame, with additional insulation  121.500000         1

                          short_desc
0   [average, thermal, transmittance]
1   [average, thermal, transmittance]
2   [average, thermal, transmittance]
3   [average, thermal, transmittance]
4   [average, thermal, transmittance]
..                                 …
81         [timber, frame, insulated]
82         [timber, frame, insulated]
```

```
83         [timber, frame, insulated]
84         [timber, frame, insulated]
85         [timber, frame, insulated]

[86 rows x 4 columns]
```

### 2.4.1 (4) - part 1:

For walls, it looks like regardless of the material used as the walls, as long as there is internal insulation, the building energy performance will be good. Buildings with low average thermal transmittance below 0.3 watt per sq meter Kelvin also have good energy performance.

[148]:
```
#create function that creates LDA and important words/topic models, given␣
↪corpus and number of topics, and number of words for each topic

def top_model_gen(corpus, num_topics, num_words):
    count_vect = CountVectorizer(stop_words=stopwords.
↪words('english'),lowercase=True)
    x_counts = count_vect.fit_transform(corpus)
    x_counts.todense()

    tfidf_transformer = TfidfTransformer()
    x_tfidf = tfidf_transformer.fit_transform(x_counts)

    lda = LDA(n_components = num_topics)
    lda_array = lda.fit_transform(x_tfidf)

    components = [lda.components_[i] for i in range(len(lda.components_))]
    features = count_vect.get_feature_names()
    important_words = [sorted(features, key = lambda x: components[j][features.
↪index(x)], reverse=True)[:num_words] for j in range(len(components))]

    return lda_array, important_words
```

[149]:
```
#we can do something similar with the roof

roof_ee_1000 = emissions_1000.groupby(['ROOF_DESCRIPTION'])['Current energy␣
↪efficiency rating'].mean().reset_index(name="EE_WALL")
roof_ee_1000
```

[149]:
```
                          ROOF_DESCRIPTION    EE_WALL
0                  (another dwelling above)  80.604520
1                 (another dwelling above)   81.062500
2                     (other premises above)  86.478814
3                    (other premises above)   85.226415
4   Average thermal transmittance 0.06 W/m²K  84.000000
..                                        …          …
```

```
74           Roof room(s), ceiling insulated   79.000000
75          Roof room(s), ceiling insulated   100.000000
76                   Roof room(s), insulated   78.000000
77                  Roof room(s), insulated    78.000000
78         Roof room(s), insulated (assumed)   82.000000

[79 rows x 2 columns]
```

[150]:
```
em_1000_roof = top_model_gen(roof_ee_1000['ROOF_DESCRIPTION'],10,4)
```

[151]:
```
em_1000_roof_array = em_1000_roof[0]
```

[152]:
```
em_1000_roof_array
```

[152]:
```
array([[0.04142136, 0.04142136, 0.04142136, 0.04142136, 0.04142136,
        0.04142136, 0.04142136, 0.04142136, 0.04142136, 0.62720779],
       [0.04142136, 0.04142136, 0.04142136, 0.04142136, 0.04142136,
        0.04142136, 0.04142136, 0.04142136, 0.04142136, 0.62720779],
       [0.05      , 0.05      , 0.55      , 0.05      , 0.05      ,
        0.05      , 0.05      , 0.05      , 0.05      , 0.05      ],
       [0.05      , 0.05      , 0.55      , 0.05      , 0.05      ,
        0.05      , 0.05      , 0.05      , 0.05      , 0.05      ],
       [0.0319562 , 0.03195621, 0.03195621, 0.25524938, 0.03195621,
        0.0319562 , 0.03195621, 0.48910095, 0.03195622, 0.03195621],
       [0.0319562 , 0.0319562 , 0.0319562 , 0.0319562 , 0.0319562 ,
        0.0319562 , 0.0319562 , 0.48905277, 0.2552976 , 0.0319562 ],
       [0.03166673, 0.03166673, 0.03166673, 0.2425279 , 0.03166673,
        0.03166673, 0.03166673, 0.50413828, 0.03166673, 0.03166673],
       [0.03166673, 0.03166673, 0.03166673, 0.2425279 , 0.03166673,
        0.03166673, 0.03166673, 0.50413828, 0.03166673, 0.03166673],
       [0.03333429, 0.03333429, 0.03333429, 0.03333429, 0.03333429,
        0.03333429, 0.03333429, 0.69999143, 0.03333429, 0.03333429],
       [0.03166673, 0.03166673, 0.03166673, 0.03166673, 0.03166673,
        0.03166673, 0.03166673, 0.71499947, 0.03166673, 0.03166673],
       [0.03166673, 0.03166673, 0.03166673, 0.03166673, 0.03166673,
        0.03166673, 0.03166673, 0.71499947, 0.03166673, 0.03166673],
       [0.03166673, 0.03166673, 0.03166673, 0.2425279 , 0.03166673,
        0.03166673, 0.03166673, 0.50413828, 0.03166673, 0.03166673],
       [0.03166673, 0.03166673, 0.03166673, 0.2425279 , 0.03166673,
        0.03166673, 0.03166673, 0.50413828, 0.03166673, 0.03166673],
       [0.03166673, 0.03166673, 0.03166673, 0.2425279 , 0.03166673,
        0.03166673, 0.03166673, 0.50413828, 0.03166673, 0.03166673],
       [0.03166673, 0.03166673, 0.03166673, 0.2425279 , 0.03166673,
        0.03166673, 0.03166673, 0.50413828, 0.03166673, 0.03166673],
       [0.0319562 , 0.0319562 , 0.0319562 , 0.0319562 , 0.03195621,
        0.0319562 , 0.0319562 , 0.71239416, 0.03195621, 0.0319562 ],
       [0.03147873, 0.03147873, 0.03147873, 0.03147873, 0.03147873,
```

```
 0.03147873, 0.03147873, 0.71669141, 0.03147873, 0.03147873],
[0.03147873, 0.03147873, 0.03147873, 0.03147873, 0.03147873,
 0.03147873, 0.03147873, 0.71669141, 0.03147873, 0.03147873],
[0.03189118, 0.03189118, 0.03189118, 0.03189118, 0.03189119,
 0.03189118, 0.23342652, 0.51144402, 0.03189119, 0.03189118],
[0.03166673, 0.03166673, 0.03166673, 0.03166673, 0.03166673,
 0.03166673, 0.03166673, 0.71499947, 0.03166673, 0.03166673],
[0.03166673, 0.03166673, 0.03166673, 0.03166673, 0.03166673,
 0.03166673, 0.03166673, 0.71499947, 0.03166673, 0.03166673],
[0.03166673, 0.03166673, 0.03166673, 0.03166673, 0.03166673,
 0.24253016, 0.03166673, 0.50413601, 0.03166673, 0.03166673],
[0.03166673, 0.03166673, 0.03166673, 0.03166673, 0.03166673,
 0.24253016, 0.03166673, 0.50413601, 0.03166673, 0.03166673],
[0.0420678 , 0.04206504, 0.62138746, 0.04207631, 0.04206504,
 0.0420726 , 0.0420659 , 0.04206504, 0.04206504, 0.04206979],
[0.03720629, 0.03720335, 0.66513425, 0.0372167 , 0.03720335,
 0.0372135 , 0.03720589, 0.03720335, 0.03720335, 0.03720998],
[0.50979413, 0.02492394, 0.02493524, 0.29079422, 0.02492209,
 0.02492975, 0.02492727, 0.02492209, 0.02492209, 0.02492918],
[0.75444399, 0.02727791, 0.02730107, 0.02729103, 0.02727637,
 0.02729069, 0.02728205, 0.02727637, 0.02727637, 0.02728416],
[0.02695006, 0.02693884, 0.02698286, 0.02693881, 0.02693606,
 0.75748256, 0.02694198, 0.02693606, 0.02693606, 0.02695671],
[0.034386  , 0.03435538, 0.69071561, 0.03436844, 0.03434198,
 0.03441344, 0.0343855 , 0.03434198, 0.03434198, 0.03434969],
[0.49716867, 0.02599238, 0.02600591, 0.29487311, 0.02599048,
 0.02599722, 0.02599482, 0.02599048, 0.02599048, 0.02599645],
[0.03443343, 0.03442805, 0.03443885, 0.03443002, 0.03442623,
 0.03443336, 0.034432  , 0.03442623, 0.03442623, 0.69012559],
[0.03443343, 0.03442805, 0.03443885, 0.03443002, 0.03442623,
 0.03443336, 0.034432  , 0.03442623, 0.03442623, 0.69012559],
[0.52504358, 0.02715757, 0.02716386, 0.02715873, 0.02715523,
 0.02716246, 0.02716132, 0.02715523, 0.02715523, 0.25768677],
[0.69947747, 0.03339228, 0.03338931, 0.0333898 , 0.03338925,
 0.03339481, 0.03339846, 0.03338925, 0.03338926, 0.0333901 ],
[0.7064968 , 0.03261273, 0.03260968, 0.03261018, 0.03260962,
 0.03261507, 0.0326162 , 0.03260962, 0.03260962, 0.03261049],
[0.0318959 , 0.03188565, 0.03188192, 0.03188245, 0.03188186,
 0.03188817, 0.71303748, 0.03188186, 0.03188186, 0.03188286],
[0.0318959 , 0.03188565, 0.03188192, 0.03188245, 0.03188186,
 0.03188817, 0.71303748, 0.03188186, 0.03188186, 0.03188286],
[0.02905976, 0.18554966, 0.02904985, 0.02905044, 0.02904978,
 0.02905674, 0.58203335, 0.02904977, 0.02904978, 0.02905088],
[0.02813923, 0.02813338, 0.02814233, 0.02814881, 0.02813093,
 0.02814937, 0.74676022, 0.02813093, 0.02813093, 0.02813385],
[0.02813923, 0.02813338, 0.02814233, 0.02814881, 0.02813093,
 0.02814937, 0.74676022, 0.02813093, 0.02813093, 0.02813385],
```

```
[0.02971777, 0.02971191, 0.02971147, 0.02971352, 0.0297088 ,
 0.02971876, 0.73258769, 0.0297088 , 0.0297088 , 0.02971249],
[0.02685174, 0.02684666, 0.02684492, 0.26633065, 0.02684349,
 0.02685069, 0.5188987 , 0.02684349, 0.0268435 , 0.02684615],
[0.032193  , 0.71038593, 0.03217329, 0.03217485, 0.03217309,
 0.03218805, 0.03219004, 0.03217309, 0.03217309, 0.03217558],
[0.032193  , 0.71038593, 0.03217329, 0.03217485, 0.03217309,
 0.03218805, 0.03219004, 0.03217309, 0.03217309, 0.03217558],
[0.59700468, 0.17045299, 0.02906601, 0.02906656, 0.02906594,
 0.02907192, 0.02907311, 0.02906594, 0.02906595, 0.0290669 ],
[0.028297  , 0.2603379 , 0.02828934, 0.02829156, 0.02828617,
 0.51334142, 0.02829579, 0.02828617, 0.02828617, 0.02828849],
[0.70868555, 0.03236955, 0.03236647, 0.03236697, 0.0323664 ,
 0.03237192, 0.03237306, 0.0323664 , 0.0323664 , 0.03236728],
[0.70868555, 0.03236955, 0.03236647, 0.03236697, 0.0323664 ,
 0.03237192, 0.03237306, 0.0323664 , 0.0323664 , 0.03236728],
[0.72956721, 0.03004516, 0.03005152, 0.03005673, 0.03004261,
 0.03005878, 0.03004949, 0.03004261, 0.03004261, 0.03004329],
[0.71042521, 0.03217626, 0.03217315, 0.03217366, 0.03217309,
 0.03217865, 0.03217982, 0.03217309, 0.03217309, 0.03217398],
[0.71042521, 0.03217626, 0.03217315, 0.03217366, 0.03217309,
 0.03217865, 0.03217982, 0.03217309, 0.03217309, 0.03217398],
[0.74536165, 0.02828827, 0.02829728, 0.02830364, 0.02828617,
 0.02830661, 0.02829374, 0.02828617, 0.02828617, 0.0282903 ],
[0.73177741, 0.02980095, 0.02980236, 0.02980517, 0.02979827,
 0.02980966, 0.02980643, 0.02979827, 0.02979827, 0.0298032 ],
[0.44695268, 0.02570048, 0.02570059, 0.34744305, 0.02569796,
 0.02570548, 0.025704  , 0.02569796, 0.02569796, 0.02569985],
[0.7064968 , 0.03261273, 0.03260968, 0.03261018, 0.03260962,
 0.03261507, 0.0326162 , 0.03260962, 0.03260962, 0.03261049],
[0.03202894, 0.03202031, 0.03201466, 0.03201502, 0.03201462,
 0.71183562, 0.03202573, 0.03201462, 0.03201462, 0.03201588],
[0.03202894, 0.03202031, 0.03201466, 0.03201502, 0.03201462,
 0.71183562, 0.03202573, 0.03201462, 0.03201462, 0.03201588],
[0.0282086 , 0.02820344, 0.028202  , 0.02820342, 0.02820001,
 0.74617342, 0.02820751, 0.02820001, 0.02820001, 0.02820158],
[0.0282086 , 0.02820344, 0.028202  , 0.02820342, 0.02820001,
 0.74617342, 0.02820751, 0.02820001, 0.02820001, 0.02820158],
[0.0268127 , 0.02680668, 0.02680321, 0.27178292, 0.02680222,
 0.51377361, 0.02681101, 0.02680222, 0.02680222, 0.02680321],
[0.04155463, 0.04154974, 0.04155681, 0.04157162, 0.04154805,
 0.62602038, 0.04155432, 0.04154805, 0.04154805, 0.04154836],
[0.03672597, 0.03672223, 0.03672958, 0.03673556, 0.03672133,
 0.66947126, 0.03672647, 0.03672133, 0.03672133, 0.03672494],
[0.03672597, 0.03672223, 0.03672958, 0.03673556, 0.03672133,
 0.66947126, 0.03672647, 0.03672133, 0.03672133, 0.03672494],
[0.03413832, 0.03413598, 0.03414007, 0.03414317, 0.03413531,
```

```
        0.69276139, 0.03413844, 0.03413531, 0.03413531, 0.03413671],
       [0.02966909, 0.02965635, 0.02965375, 0.58090885, 0.02964998,
        0.18183641, 0.0296679 , 0.02964998, 0.02964999, 0.02965769],
       [0.03009211, 0.03008859, 0.03009088, 0.55586327, 0.20341244,
        0.03009738, 0.0300922 , 0.03008725, 0.03008726, 0.03008861],
       [0.03187244, 0.03186888, 0.0318712 , 0.71316557, 0.03186752,
        0.03187794, 0.03187253, 0.03186752, 0.03186752, 0.0318689 ],
       [0.03187244, 0.03186888, 0.0318712 , 0.71316557, 0.03186752,
        0.03187794, 0.03187253, 0.03186752, 0.03186752, 0.0318689 ],
       [0.03188458, 0.03188118, 0.0318842 , 0.71305208, 0.03188001,
        0.03189092, 0.03188485, 0.03188   , 0.03188001, 0.03188217],
       [0.03188458, 0.03188118, 0.0318842 , 0.71305208, 0.03188001,
        0.03189092, 0.03188485, 0.03188   , 0.03188001, 0.03188217],
       [0.03011155, 0.03009494, 0.0300898 , 0.59066085, 0.03008631,
        0.16857665, 0.03010972, 0.03008631, 0.03008631, 0.03009755],
       [0.0390783 , 0.03907634, 0.03907917, 0.03908158, 0.03907567,
        0.64830365, 0.03907817, 0.03907567, 0.03907567, 0.03907579],
       [0.03508422, 0.03505385, 0.03504389, 0.03505692, 0.03503952,
        0.03510842, 0.0350836 , 0.03503952, 0.03503952, 0.68445054],
       [0.03672145, 0.03671363, 0.03671399, 0.03671687, 0.03671019,
        0.66956516, 0.03672151, 0.03671019, 0.03671019, 0.03671681],
       [0.03424053, 0.03424012, 0.03424301, 0.44931206, 0.27676012,
        0.03424334, 0.03424046, 0.03424012, 0.03424012, 0.03424012],
       [0.03424053, 0.03424012, 0.03424301, 0.44931206, 0.27676012,
        0.03424334, 0.03424046, 0.03424012, 0.03424012, 0.03424012],
       [0.03680213, 0.03680173, 0.03680424, 0.66877848, 0.03680173,
        0.03680444, 0.03680206, 0.03680173, 0.03680173, 0.03680173],
       [0.03680213, 0.03680173, 0.03680424, 0.66877848, 0.03680173,
        0.03680444, 0.03680206, 0.03680173, 0.03680173, 0.03680173],
       [0.03354623, 0.0335454 , 0.03354888, 0.69807995, 0.0335454 ,
        0.03354976, 0.03354651, 0.0335454 , 0.0335454 , 0.03354707]])
```

[153]:
```
em_1000_roof_words = em_1000_roof[1]
```

[154]:
```
em_1000_roof_words
```

[154]:
```
[['insulation', 'mm', 'loft', 'pitched'],
 ['250', 'loft', 'mm', 'insulation'],
 ['flat', 'premises', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['ceiling', 'roof', 'room', 'insulated'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['200', 'insulation', 'pitched', 'mm'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['08', 'm²k', 'average', 'thermal'],
 ['limited', 'another', 'dwelling', 'flat']]
```

[155]:
```
#creating a function that returns the topic number from the LDA array
def topic_num_column(dataframe, array):
    description_no = []
    for i in range(len(dataframe)):
        list_array = list(array[i])
        max_num = list_array.index(max(list_array))
        description_no.append(max_num)
    return description_no
```

[156]:
```
#creating a function that returns the topic model from the topic number
def short_desc_column(topic_num_list, words):
    short_desc = []
    for i in range(len(topic_num_list)):
        description = words[topic_num_list[i]]
        short_desc.append(description)
    return short_desc
```

[157]:
```
roof_topic = topic_num_column(roof_ee_1000, em_1000_roof_array)
roof_topic
```

[157]:
```
[9,
 9,
 2,
 2,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 2,
 2,
 0,
 0,
```

```
5,
2,
0,
9,
9,
0,
0,
0,
6,
6,
6,
6,
6,
6,
6,
1,
1,
0,
5,
0,
0,
0,
0,
0,
0,
0,
0,
0,
5,
5,
5,
5,
5,
5,
5,
5,
5,
3,
3,
3,
3,
3,
3,
3,
5,
9,
5,
```

```
 3,
 3,
 3,
 3,
 3]
```

[158]:
```
len(roof_topic)
```

[158]:
```
79
```

[159]:
```
roof_words = short_desc_column(roof_topic, em_1000_roof_words)
roof_words
```

[159]:
```
[['limited', 'another', 'dwelling', 'flat'],
 ['limited', 'another', 'dwelling', 'flat'],
 ['flat', 'premises', 'insulated', 'assumed'],
 ['flat', 'premises', 'insulated', 'assumed'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['average', 'thermal', 'transmittance', 'm²k'],
 ['flat', 'premises', 'insulated', 'assumed'],
 ['flat', 'premises', 'insulated', 'assumed'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['flat', 'premises', 'insulated', 'assumed'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['limited', 'another', 'dwelling', 'flat'],
 ['limited', 'another', 'dwelling', 'flat'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
```

```
 ['insulation', 'mm', 'loft', 'pitched'],
 ['200', 'insulation', 'pitched', 'mm'],
 ['200', 'insulation', 'pitched', 'mm'],
 ['200', 'insulation', 'pitched', 'mm'],
 ['200', 'insulation', 'pitched', 'mm'],
 ['200', 'insulation', 'pitched', 'mm'],
 ['200', 'insulation', 'pitched', 'mm'],
 ['200', 'insulation', 'pitched', 'mm'],
 ['250', 'loft', 'mm', 'insulation'],
 ['250', 'loft', 'mm', 'insulation'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['insulation', 'mm', 'loft', 'pitched'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['limited', 'another', 'dwelling', 'flat'],
 ['pitched', 'insulated', '400', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed'],
 ['roof', 'room', 'insulated', 'assumed']]
```

[160]:

```
len(roof_words)
```

[160]: 79

[161]:
```
roof_ee_1000['topic_number']= roof_topic
roof_ee_1000['short_description'] = roof_words
roof_ee_1000.head(10)
```

[161]:
```
                         ROOF_DESCRIPTION    EE_WALL  topic_number  \
0              (another dwelling above)  80.604520             9
1             (another dwelling above)   81.062500             9
2                (other premises above)  86.478814             2
3               (other premises above)   85.226415             2
4  Average thermal transmittance 0.06 W/m²K  84.000000             7
5  Average thermal transmittance 0.08 W/m²K  86.000000             7
6  Average thermal transmittance 0.09 W/m²K  86.521277             7
7 Average thermal transmittance 0.09 W/m²K   84.285714             7
8   Average thermal transmittance 0.1 W/m²K  96.681818             7
9  Average thermal transmittance 0.10 W/m²K  90.030303             7

                       short_description
0      [limited, another, dwelling, flat]
1      [limited, another, dwelling, flat]
2    [flat, premises, insulated, assumed]
3    [flat, premises, insulated, assumed]
4  [average, thermal, transmittance, m²k]
5  [average, thermal, transmittance, m²k]
6  [average, thermal, transmittance, m²k]
7  [average, thermal, transmittance, m²k]
8  [average, thermal, transmittance, m²k]
9  [average, thermal, transmittance, m²k]
```

[162]:
```
roof_ee_1000 = roof_ee_1000.sort_values(by='EE_WALL').reset_index(drop=True)
roof_ee_1000
```

[162]:
```
                                     ROOF_DESCRIPTION     EE_WALL  \
0   Pitched, insulated (assumed) | Pitched, no ins…   11.000000
1   Pitched, 200 mm loft insulation | Pitched, no …   39.000000
2                      Pitched, no insulation (assumed)   56.000000
3                   Flat, limited insulation (assumed)    59.000000
4   Pitched, 300 mm loft insulation | Pitched, ins…   59.000000
..                                                  …         …
74           Average thermal transmittance 0.12 W/m²K  102.823529
75           Average thermal transmittance 0.15 W/m²K  104.375000
76                    Pitched, 400+ mm loft insulation  104.600000
77          Average thermal transmittance 0.11 W/m²K   105.500000
78          Average thermal transmittance 0.10 W/m²K   107.000000

    topic_number                       short_description
```

```
0              3        [roof, room, insulated, assumed]
1              6          [200, insulation, pitched, mm]
2              5      [pitched, insulated, 400, assumed]
3              9      [limited, another, dwelling, flat]
4              0         [insulation, mm, loft, pitched]
..           …                                      …
74             7  [average, thermal, transmittance, m²k]
75             7  [average, thermal, transmittance, m²k]
76             5      [pitched, insulated, 400, assumed]
77             7  [average, thermal, transmittance, m²k]
78             7  [average, thermal, transmittance, m²k]

[79 rows x 4 columns]
```

### 2.4.2 For (4) - part 2:

It looks like if there is insulation, pitched with loft insulation of more than 200mm, and if the average thermal transmittance is low, then the building is expected to have good energy performance. These qualities ought to be recommended for any building where possible.

## 2.5 Algorithm Challenge 5: Build an algorithm that takes as input the characteristics of a building (any field of the dataset except those related to costs) and outputs the total cost of energy of the building over a 3-year period - (15 points)

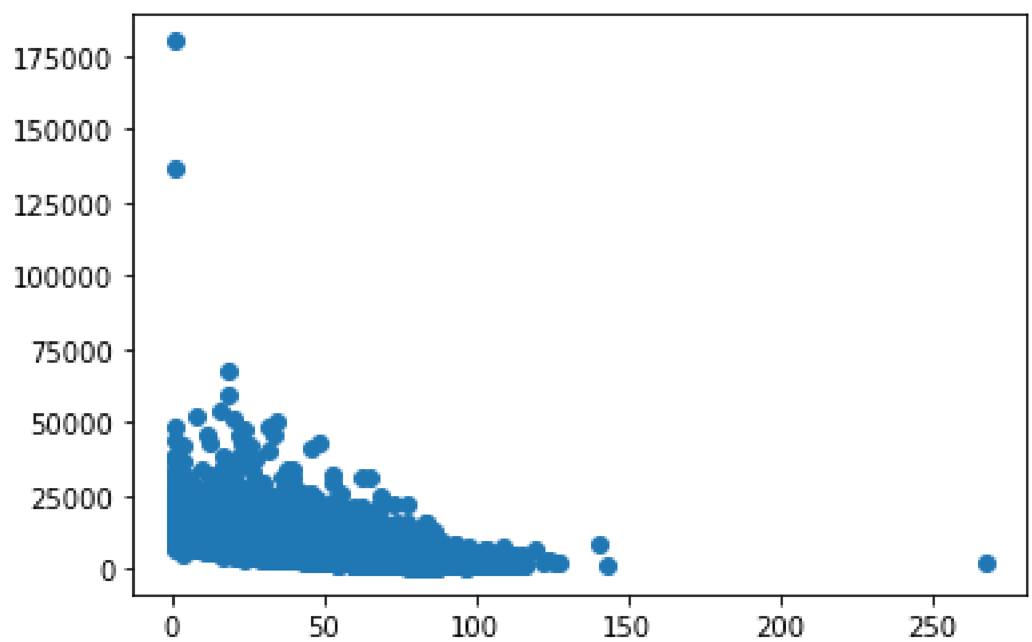
[163]:
```
dataset.columns
```

[163]:
```
Index(['Property_UPRN', 'Postcode', 'POST_TOWN', 'Date of Assessment',
       'Primary Energy Indicator (kWh/m²/year)', 'Total floor area (m²)',
       'Current energy efficiency rating',
       'Current energy efficiency rating band',
       'Potential Energy Efficiency Rating',
       'Potential energy efficiency rating band',
       'Current Environmental Impact Rating',
       'Current Environmental Impact Rating Band',
       'Potential Environmental Impact Rating',
       'Potential Environmental Impact Rating Band',
       'CO2 Emissions Current Per Floor Area (kg.CO2/m²/yr)',
       'WALL_DESCRIPTION', 'WALL_ENERGY_EFF', 'ROOF_DESCRIPTION',
       'ROOF_ENERGY_EFF', 'FLOOR_DESCRIPTION', 'FLOOR_ENERGY_EFF',
       'FLOOR_ENV_EFF', 'WINDOWS_DESCRIPTION', 'WINDOWS_ENERGY_EFF',
       'WINDOWS_ENV_EFF', 'MAINHEAT_DESCRIPTION', 'MAINHEAT_ENERGY_EFF',
       'MAINHEAT_ENV_EFF', 'MAINHEATCONT_DESCRIPTION', 'MAINHEATC_ENERGY_EFF',
       'MAINHEATC_ENV_EFF', 'HOT_WATER_ENERGY_EFF', 'HOT_WATER_ENV_EFF',
       'LIGHTING_DESCRIPTION', 'LIGHTING_ENERGY_EFF', 'LIGHTING_ENV_EFF',
       'Current Emissions (T.CO2/yr)',
       'Potential Reduction in Emissions (T.CO2/yr)',
```

```
'Total current energy costs over 3 years (£)',
'Current heating costs over 3 years (£)',
'Potential heating costs over 3 years (£)',
'Current hot water costs over 3 years (£)',
'Potential hot water costs over 3 years (£)',
'Current lighting costs over 3 years (£)',
'Potential lighting costs over 3 years (£)',
'Part 1 Construction Age Band', 'Built Form', 'Property Type',
'heat_savings', 'hot_water_save', 'AGG_RATING', 'EE_PRODUCT',
'ROOF_RATING', 'ROOF_EE_PRODUCT', 'est_build_year', 'build_age'],
dtype='object')
```

5. The column to focus on would be "Total current energy costs over 3 years (£)". That would be the dependent variable. The independent variable could be 'Current energy efficiency rating'. The proposal here is to create a linear regression model.

[164]:
```
plt.scatter(dataset['Current energy efficiency rating'],  dataset['Total␣
↪current energy costs over 3 years (£)'])
plt.show()
```




[165]:
```
model_x = dataset['Current energy efficiency rating']
model_y = dataset['Total current energy costs over 3 years (£)']
```

[166]:
```
model_x = np.array(model_x)
model_y = np.array(model_y)
```

```
model_x = model_x.reshape(-1,1)
model_y = model_y.reshape(-1,1)
```

[167]:
```
model_eff_costs = LinearRegression().fit(model_x, model_y)
```

[168]:
```
model_eff_costs.coef_
```

[168]:
```
array([[-108.03900306]])
```

[169]:
```
model_eff_costs.intercept_
```

[169]:
```
array([10166.4758954])
```

Generally speaking, the lower the efficiency score, the lower the total energy costs over 3 years. Every improvement in energy efficiency saves 108 pounds in energy costs over 3 years.

### 2.6 Algorithm Challenge 6: Build an algorithm that takes as input the characteristics of a building (any field in the dataset) and outputs recommendations on which elements of the house should be modified to most effectively decrease the total energy cost of the building over a 3-year period

We have shown how the walls and roofs of the buildings could be improved to improve their energy performance in previous sections - see response to algorithm challenges 1, 2, and 4. Using aggregations and LDA, we show that insulation of walls and roofs regardless of wall materials can boost energy efficiency and lower energy costs over 3 years, as shown in algorithm challenge 5.

An appropriate algorithm would build on the results of 1, 2, and 4 to provide guidance on how to boost energy performance as defined by energy efficiency.

Adding insulation appears to be the key to improving energy performance as defined by energy efficiency.

[ ]:
```
```

[ ]:
```
```