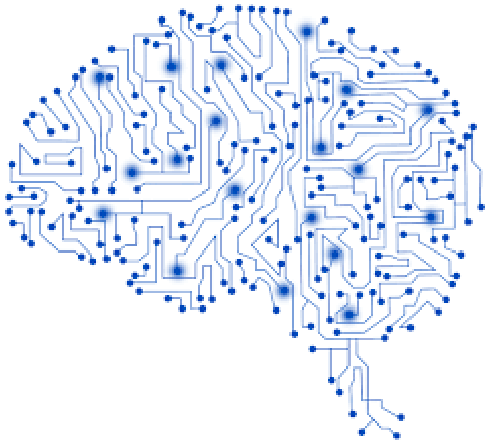
University of Minho
School of Engineering



Dados e Aprendizagem Automática

Feature Engineering

DAA @ MEI-1º/MiEI-4º – 1º Semestre

Bruno Fernandes, César Analide, Dalila Alves, Filipa Ferraz, Victor Alves

Part V

Contents

Feature Engineering

Feature Engineering

- Exercise:
  - Dataset: table with information regarding the *incidents* on the road with 5000 entries and 13 features, including:
    - city_name
    - magnitude_of_delay
    - delay_in_seconds
    - affected_roads
    - record_date
    - luminosity
    - avg_temperature
    - avg_atm_pressure
    - avg_humidity
    - avg_wind_speed
    - avg_precipitation
    - avg_rain
    - incidents

Feature Engineering

5 Get the data

```
data = pd.read_csv('incidents.csv')
```

```
data.columns
```

```
Index(['city_name', 'magnitude_of_delay', 'delay_in_seconds', 'affected_roads',
       'record_date', 'luminosity', 'avg_temperature', 'avg_atm_pressure',
       'avg_humidity', 'avg_wind_speed', 'avg_precipitation', 'avg_rain',
       'incidents'],
      dtype='object')
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   city_name           5000 non-null   object
 1   magnitude_of_delay  5000 non-null   object
 2   delay_in_seconds    5000 non-null   int64
 3   affected_roads      4915 non-null   object
 4   record_date         5000 non-null   object
 5   luminosity          5000 non-null   object
 6   avg_temperature     5000 non-null   float64
 7   avg_atm_pressure    5000 non-null   float64
 8   avg_humidity        5000 non-null   float64
 9   avg_wind_speed      5000 non-null   float64
 10  avg_precipitation   5000 non-null   float64
 11  avg_rain            5000 non-null   object
 12  incidents           2972 non-null   object
dtypes: float64(5), int64(1), object(7)
memory usage: 507.9+ KB
```

```
data.head()
```

	city_name	magnitude_of_delay	delay_in_seconds	affected_roads	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_precipitation	avg_rain	incidents
0	Guimaraes	UNDEFINED	0	,	2021-03-15 23:00	DARK	12.0	1013.0	70.0	1.0	0.0	Sem Chuva	NaN
1	Guimaraes	UNDEFINED	385	N101,	2021-12-25 18:00	DARK	12.0	1007.0	91.0	1.0	0.0	Sem Chuva	NaN
2	Guimaraes	UNDEFINED	69	,	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	0.0	Sem Chuva	Low
3	Guimaraes	MAJOR	2297	N101,R206,N105,N101,N101,N101,N101,N101,N101,N...	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	0.0	Sem Chuva	Very_High
4	Guimaraes	UNDEFINED	0	N101,N101,N101,N101,N101,	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	0.0	Sem Chuva	High

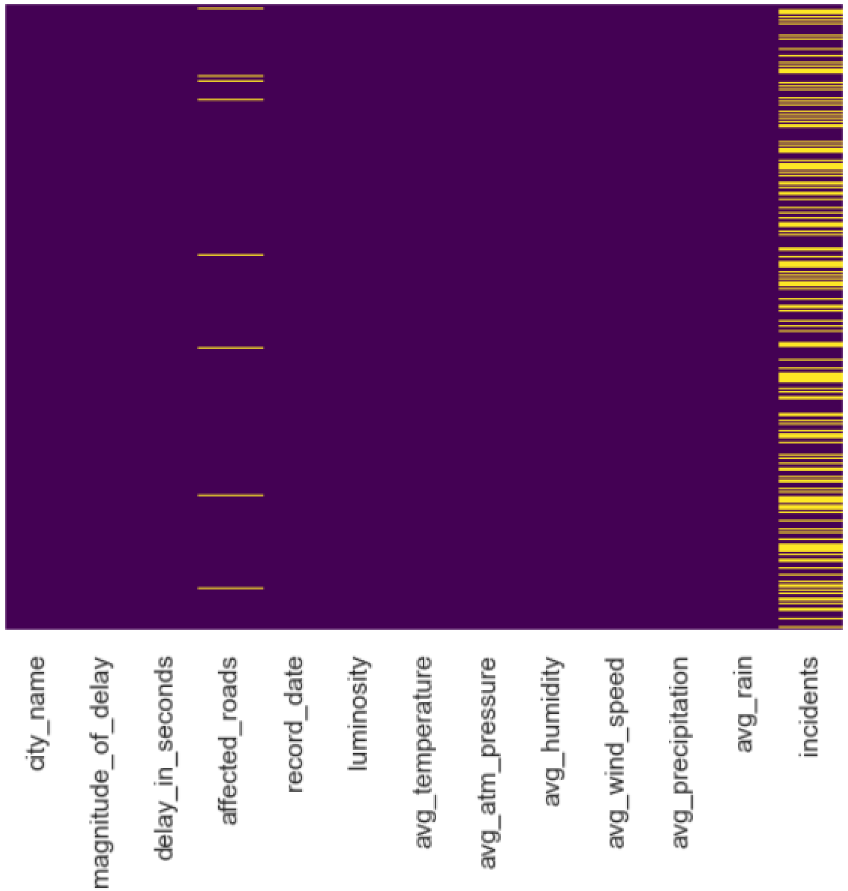
Feature Engineering

6 Handling missing data and possible data transformations

- Remove missing values, outliers, and unnecessary rows/ columns
- Check and impute null values
- Check Imbalanced data
- Re-indexing and reformatting our data

1. Missing Values

```
sns.heatmap(data.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```




```
data.isnull().sum()
```

```
city_name               0
magnitude_of_delay      0
delay_in_seconds        0
affected_roads         85
record_date             0
luminosity              0
avg_temperature         0
avg_atm_pressure        0
avg_humidity            0
avg_wind_speed          0
avg_precipitation       0
avg_rain                0
incidents            2028
dtype: int64
```

Feature Engineering

7 Drop or fill

Let's verify how the data is presented in the feature *affected_roads*

```
data['affected_roads'].head()
```

```
0                                          ,
1                                      N101,
2                                          ,
3    N101,R206,N105,N101,N101,N101,N101,N101,N101,N...
4                        N101,N101,N101,N101,N101,
Name: affected_roads, dtype: object
```

```
data[data['affected_roads'].isnull()]
```

	city_name	magnitude_of_delay	delay_in_seconds	affected_roads	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_precipitation	avg_rain	incidents
29	Guimaraes	UNDEFINED	64	NaN	2021-01-22 09:00	LIGHT	8.0	1012.0	91.0	4.0	0.0	Sem Chuva	Medium
76	Guimaraes	UNDEFINED	223	NaN	2021-01-29 08:00	LIGHT	11.0	1022.0	92.0	1.0	0.0	Sem Chuva	High
79	Guimaraes	MAJOR	80	NaN	2021-12-24 21:00	DARK	11.0	1004.0	92.0	0.0	0.0	Sem Chuva	NaN
91	Guimaraes	UNDEFINED	52	NaN	2021-03-02 13:00	LIGHT	13.0	1024.0	78.0	2.0	0.0	Sem Chuva	Low
109	Guimaraes	UNDEFINED	139	NaN	2021-12-27 13:00	LIGHT	15.0	1014.0	88.0	5.0	0.0	Sem Chuva	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4785	Guimaraes	MAJOR	298	NaN	2021-12-22 13:00	LIGHT	16.0	1015.0	71.0	3.0	0.0	Sem Chuva	NaN
4811	Guimaraes	UNDEFINED	96	NaN	2021-03-11 15:00	LIGHT	13.0	1025.0	89.0	3.0	0.0	chuva fraca	Medium
4838	Guimaraes	UNDEFINED	36	NaN	2021-03-10 13:00	LIGHT	14.0	1025.0	65.0	2.0	0.0	Sem Chuva	Low
4854	Guimaraes	UNDEFINED	233	NaN	2021-01-29 20:00	DARK	11.0	1017.0	92.0	1.0	0.0	Sem Chuva	High
4910	Guimaraes	UNDEFINED	324	NaN	2021-02-03 08:00	LIGHT	10.0	1012.0	90.0	2.0	0.0	Sem Chuva	Low

85 rows × 13 columns

Feature Engineering

Copy of the data to experiment the options

```
data_m1 = data.copy()
data_m2 = data.copy()
```

a) Drop

```
data_m1.drop(['affected_roads'], axis = 1, inplace = True)
data_m1.head()
```

	city_name	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_precipitation	avg_rain	incidents
0	Guimaraes	UNDEFINED	0	2021-03-15 23:00	DARK	12.0	1013.0	70.0	1.0	0.0	Sem Chuva	NaN
1	Guimaraes	UNDEFINED	385	2021-12-25 18:00	DARK	12.0	1007.0	91.0	1.0	0.0	Sem Chuva	NaN
2	Guimaraes	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	0.0	Sem Chuva	Low
3	Guimaraes	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	0.0	Sem Chuva	Very_High
4	Guimaraes	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	0.0	Sem Chuva	High

Feature Engineering

b) Fill with zero

```
data_m2.fillna(0, inplace = True)
data_m2.head()
```

	city_name	magnitude_of_delay	delay_in_seconds	affected_roads	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_precipitation	avg_rain	incidents
0	Guimaraes	UNDEFINED	0	,	2021-03-15 23:00	DARK	12.0	1013.0	70.0	1.0	0.0	Sem Chuva	0
1	Guimaraes	UNDEFINED	385	N101,	2021-12-25 18:00	DARK	12.0	1007.0	91.0	1.0	0.0	Sem Chuva	0
2	Guimaraes	UNDEFINED	69	,	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	0.0	Sem Chuva	Low
3	Guimaraes	MAJOR	2297	N101,R206,N105,N101,N101,N101,N101,N101,N101,N...	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	0.0	Sem Chuva	Very_High
4	Guimaraes	UNDEFINED	0	N101,N101,N101,N101,N101,	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	0.0	Sem Chuva	High

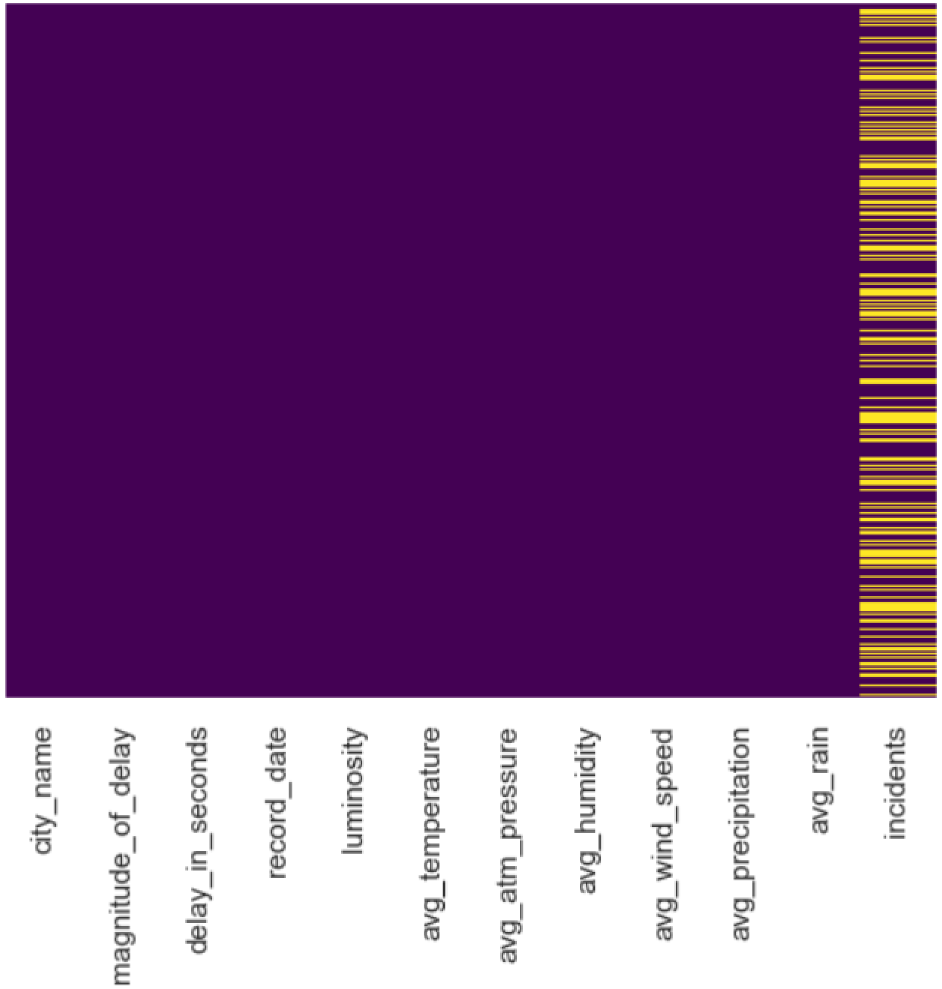
We need to choose one of the options to keep going. We will choose to drop the column since it does not bring added value to our goal.

```
data.drop(['affected_roads'], axis = 1, inplace = True)
```

Let's see if there are still missing values

Feature Engineering


```
sns.heatmap(data.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```




```
data.isnull().sum()
```

```
city_name               0
magnitude_of_delay      0
delay_in_seconds        0
record_date             0
luminosity              0
avg_temperature         0
avg_atm_pressure        0
avg_humidity            0
avg_wind_speed          0
avg_precipitation       0
avg_rain                0
incidents            2028
dtype: int64
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   city_name           5000 non-null   object
 1   magnitude_of_delay  5000 non-null   object
 2   delay_in_seconds    5000 non-null   int64
 3   record_date         5000 non-null   object
 4   luminosity          5000 non-null   object
 5   avg_temperature     5000 non-null   float64
 6   avg_atm_pressure    5000 non-null   float64
 7   avg_humidity        5000 non-null   float64
 8   avg_wind_speed      5000 non-null   float64
 9   avg_precipitation   5000 non-null   float64
 10  avg_rain            5000 non-null   object
 11  incidents           2972 non-null   object
dtypes: float64(5), int64(1), object(6)
memory usage: 468.9+ KB
```

Feature Engineering


```
data.head()
```

	city_name	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_precipitation	avg_rain	incidents
0	Guimaraes	UNDEFINED	0	2021-03-15 23:00	DARK	12.0	1013.0	70.0	1.0	0.0	Sem Chuva	NaN
1	Guimaraes	UNDEFINED	385	2021-12-25 18:00	DARK	12.0	1007.0	91.0	1.0	0.0	Sem Chuva	NaN
2	Guimaraes	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	0.0	Sem Chuva	Low
3	Guimaraes	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	0.0	Sem Chuva	Very_High
4	Guimaraes	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	0.0	Sem Chuva	High

There are features that are of the type *object*: *city_name*, *magnitude_of_delay*, *record_date*, *luminosity*, *avg_rain* and *incidents*.

Let's see how many different values each feature has.

```
data.nunique()
```

```
city_name                 1
magnitude_of_delay        3
delay_in_seconds       1186
record_date            5000
luminosity                3
avg_temperature          35
avg_atm_pressure         36
avg_humidity             83
avg_wind_speed           11
avg_precipitation         1
avg_rain                  4
incidents                 4
dtype: int64
```

Feature Engineering

The features *city_name* and *avg_precipitation* have only one value. We will start with *avg_precipitation*:

```
data.nunique()
```

```
city_name                1
magnitude_of_delay       3
delay_in_seconds      1186
record_date           5000
luminosity               3
avg_temperature         35
avg_atm_pressure        36
avg_humidity            83
avg_wind_speed          11
avg_precipitation        1
avg_rain                 4
incidents                4
dtype: int64
```

```
data['avg_precipitation'].nunique()
```

```
1
```

```
data['avg_precipitation'].describe()
```

```
count    5000.0
mean        0.0
std         0.0
min         0.0
25%         0.0
50%         0.0
75%         0.0
max         0.0
Name: avg_precipitation, dtype: float64
```

Since 0 is the unique value of *avg_precipitation* and all entries have the same value, we will drop this feature.

```
data.drop(['avg_precipitation'], axis = 1, inplace = True)
```

Feature Engineering

2. Handling categoric data

Feature *city_name*

```
data['city_name'].head()
```

```
0    Guimaraes
1    Guimaraes
2    Guimaraes
3    Guimaraes
4    Guimaraes
Name: city_name, dtype: object
```

The unique value of *city_name* is *Guimarães*. We can drop this feature as well.

```
data.drop('city_name',axis=1,inplace=True)
data.dropna(inplace=True)
```

Let's see the feature *incidents*:

```
print(data['incidents'].value_counts())
```

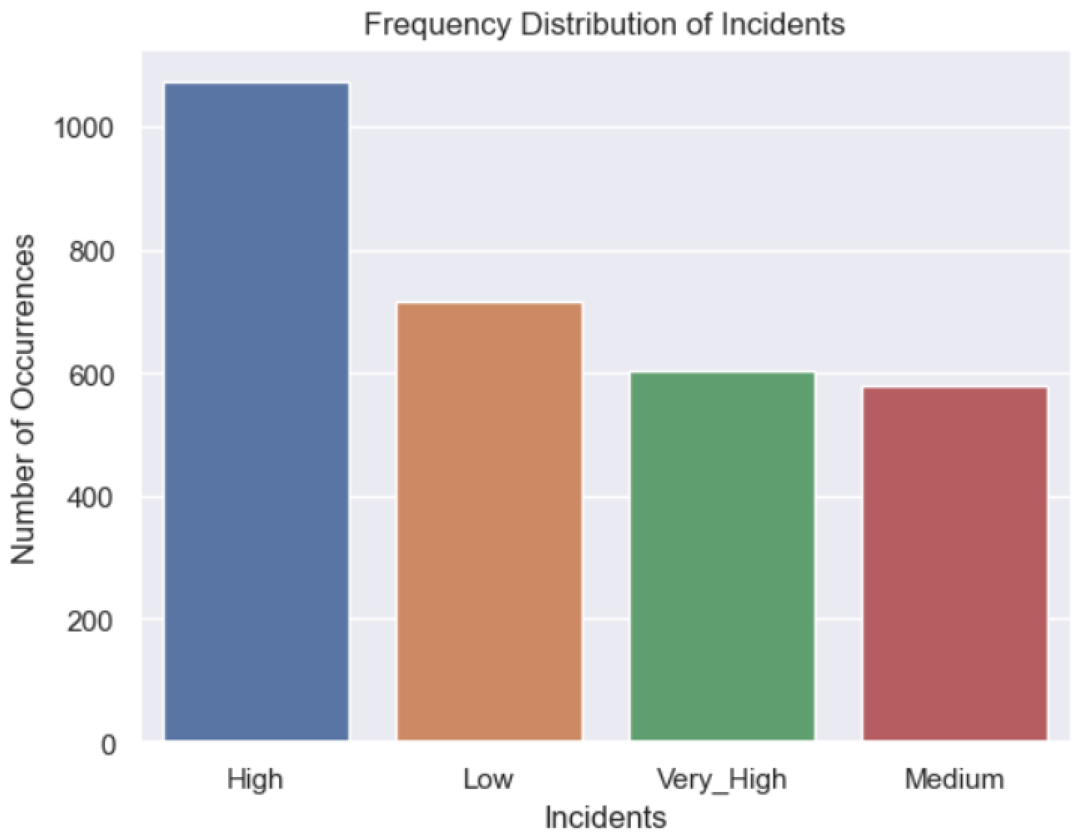
```
incidents
High         1073
Low           718
Very_High     603
Medium        578
Name: count, dtype: int64
```

```
print(data['incidents'].value_counts().count())
```

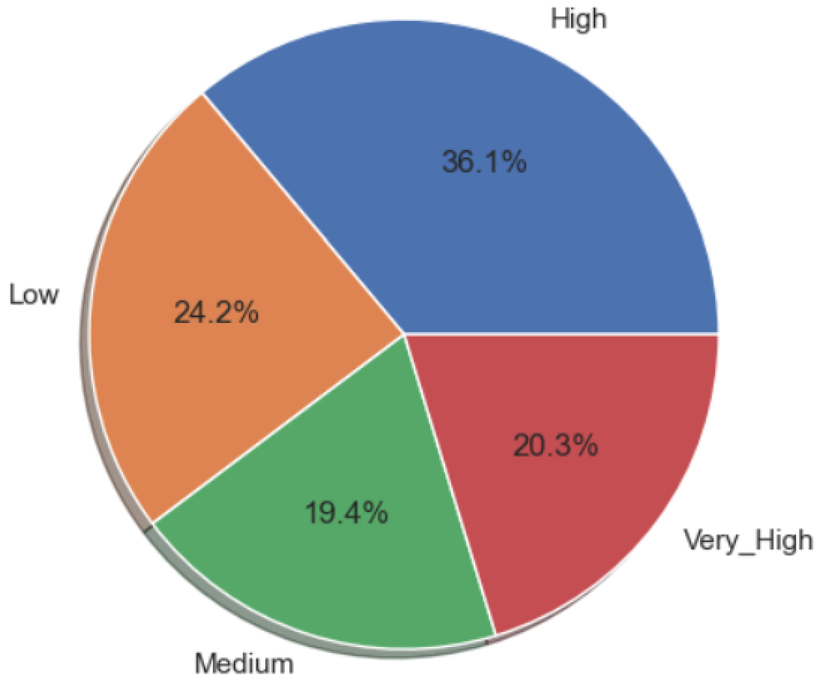
```
4
```

Feature Engineering


```
incidents_count = data['incidents'].value_counts()
sns.set(style="darkgrid")
sns.barplot(x=incidents_count.index, y=incidents_count.values)
plt.title('Frequency Distribution of Incidents')
plt.ylabel('Number of Occurrences', fontsize=12)
plt.xlabel('Incidents', fontsize=12)
plt.show()
```




```
labels = data['incidents'].astype('category').cat.categories.tolist()
counts = data['incidents'].value_counts()
sizes = [counts[var_cat] for var_cat in labels]
fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=True)
ax1.axis('equal')
plt.show()
```



We have several options how to deal with qualitative data:

Feature Engineering

a) Replace Values

Again, we are using data copies to experiment all options.

```
data_r1=data.copy()

data_r1.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents
2	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	Low
3	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	Very_High
4	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	High
5	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	Medium
6	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	Low

We need to create a dictionary assigning the string to a numeric value:

None - 0, Low - 1, Medium - 2, High - 3, Very_High - 4

```
replace_map = {'incidents': {'None': 0, 'Low': 1, 'Medium': 2, 'High': 3, 'Very_High': 4}}
```

We can create a replacement map in other way:

```
labels = data_r1['incidents'].astype('category').cat.categories.tolist()
replace_map_comp = {'incidents' : {k: v for k,v in zip(labels,list(range(1,len(labels)+1)))}}
print(replace_map_comp)
```

```
{'incidents': {'High': 1, 'Low': 2, 'Medium': 3, 'Very_High': 4}}
```

Feature Engineering


```
data_r1.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents
2	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	Low
3	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	Very_High
4	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	High
5	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	Medium
6	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	Low

Now we need to replace with the new values:

```
data_r1.replace(replace_map_comp, inplace=True)
data_r1.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents
2	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	2
3	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	4
4	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	1
5	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	3
6	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	2

```
print(data_r1['incidents'].dtypes)
```

```
int64
```

Feature Engineering

b) Label encoding

```
data_r2=data.copy()

data_r2.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents
2	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	Low
3	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	Very_High
4	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	High
5	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	Medium
6	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	Low

```
print(data_r2.dtypes)
```

```
magnitude_of_delay     object
delay_in_seconds        int64
record_date            object
luminosity             object
avg_temperature       float64
avg_atm_pressure      float64
avg_humidity          float64
avg_wind_speed        float64
avg_rain               object
incidents              object
dtype: object
```

Feature Engineering

Similar to the previous examples, each string will be assigned a number. Instead of replacing the values under the column *incidents*, we are going to create a new colum to each created label.

```
data_r2['None'] = np.where(data_r2['incidents'].str.contains('None'), 1, 0)
data_r2.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents	None
2	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	Low	0
3	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	Very_High	0
4	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	High	0
5	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	Medium	0
6	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	Low	0

To complete the process, it is needed to replicate for each label and then drop the column *incidents*.

Feature Engineering

Let's see another way to label encoding. This uses the *LabelEncoder* from *sklearn*.

```
data_r2_skl = data.copy()
data_r22=data.copy()

from sklearn.preprocessing import LabelEncoder

lb_make = LabelEncoder()
data_r2_skl['incidents_code'] = lb_make.fit_transform(data_r22['incidents'])

data_r2_skl.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents	incidents_code
2	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	Low	1
3	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	Very_High	3
4	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	High	0
5	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	Medium	2
6	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	Low	1

It creates a new column, *incidents_code*, with the labels assigned to feature *incidents*. The numeric values were assigned randomly, being the crescent order not apllicable to the meaning of the qualifying words.

Feature Engineering

c) One-Hot encoding

This alternative uses *LabelBinarizer* of *sklearn* and creates a matrix with bits regarding each label.

```
data_r3 = data.copy()

from sklearn.preprocessing import LabelBinarizer

lb = LabelBinarizer()
lb_results = lb.fit_transform(data_r3['incidents'])
lb_results_df = pd.DataFrame(lb_results, columns=lb.classes_)

lb_results_df.head()
```

	High	Low	Medium	Very_High
0	0	1	0	0
1	0	0	0	1
2	1	0	0	0
3	0	0	1	0
4	0	1	0	0

```
result_df = pd.concat([data_r3, lb_results_df], axis=1)
```

```
result_df.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents	High	Low	Medium	Very_High
2	UNDEFINED	69.0	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	Low	1.0	0.0	0.0	0.0
3	MAJOR	2297.0	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	Very_High	0.0	0.0	1.0	0.0

Feature Engineering

d) Binary Encoding

Similar to the previous technique, it creates a matrix of the status of the values, but this time with binary values. See the comparison between techniques below:

Level	"Decimal encoding"	Binary encoding	One-Hot encoding
None	0	000	000001
Low	1	001	000010
Medium	2	010	000100
High	3	011	001000
Very_High	4	100	010000

For this technique it is needed to have the *category_encoders* installed: `!pip install category_encoders`

```
data_r4 = data.copy()

import category_encoders as ce

encoder = ce.BinaryEncoder(cols=['incidents'])
df_binary = encoder.fit_transform(data_r4)

df_binary.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents_0	incidents_1	incidents_2
2	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	0	0	1
3	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	0	1	0
4	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	0	1	1
5	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	1	0	0
6	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	0	0	1

Feature Engineering

e) Backward difference encoding

The values are normalized in the range of -1 to 1.

```
data_r5 = data.copy()

encoder = ce.BackwardDifferenceEncoder(cols=['incidents'])
df_bd = encoder.fit_transform(data_r5)

df_bd.head()
```

	intercept	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents_0	incidents_1	incidents_2
2	1	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	-0.75	-0.5	-0.25
3	1	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	0.25	-0.5	-0.25
4	1	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	0.25	0.5	-0.25
5	1	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	0.25	0.5	0.75
6	1	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	-0.75	-0.5	-0.25

Feature Engineering

f) Factorize

This technique encodes the object as an enumerated type or categorical variable.

```
data_r6 = data.copy()
```

```
data_r6['incidents'] = pd.factorize(data_r6['incidents'])[0] + 1
data_r6.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents
2	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	1
3	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	2
4	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	3
5	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	4
6	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	1

We will choose the factorize technique to keep going.

```
data['incidents'] = pd.factorize(data['incidents'])[0] + 1
data.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents
2	UNDEFINED	69	2021-03-12 15:00	LIGHT	14.0	1025.0	64.0	0.0	Sem Chuva	1
3	MAJOR	2297	2021-09-29 09:00	LIGHT	15.0	1028.0	75.0	1.0	Sem Chuva	2
4	UNDEFINED	0	2021-06-13 11:00	LIGHT	27.0	1020.0	52.0	1.0	Sem Chuva	3
5	UNDEFINED	0	2021-12-07 23:00	DARK	9.0	1015.0	94.0	0.0	Sem Chuva	4
6	UNDEFINED	0	2021-12-05 05:00	DARK	8.0	1026.0	87.0	1.0	Sem Chuva	1

Feature Engineering

Other option would it be to filter the *NaN* values when reading the CSV file:

```
data = pd.read_csv('incidents.csv', na_filter=False)
```

Regarding the features *magnitude_delay*, *luminosity* and *avg_rain*, we will factorize for now.

```
data['magnitude_of_delay'] = pd.factorize(data['magnitude_of_delay'])[0] + 1
data['luminosity'] = pd.factorize(data['luminosity'])[0] + 1
data['avg_rain'] = pd.factorize(data['avg_rain'])[0] + 1

data.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents
2	1	69	2021-03-12 15:00	1	14.0	1025.0	64.0	0.0	1	1
3	2	2297	2021-09-29 09:00	1	15.0	1028.0	75.0	1.0	1	2
4	1	0	2021-06-13 11:00	1	27.0	1020.0	52.0	1.0	1	3
5	1	0	2021-12-07 23:00	2	9.0	1015.0	94.0	0.0	1	4
6	1	0	2021-12-05 05:00	2	8.0	1026.0	87.0	1.0	1	1

Feature Engineering

3. Handling dates

Datetime Properties and Methods (https://pandas.pydata.org/pandas-docs/version/0.23/api.html#datetimelike-properties)

```
data_dt = data.copy()
```

```
data_dt['record_date'].head()
```

```
2    2021-03-12 15:00
3    2021-09-29 09:00
4    2021-06-13 11:00
5    2021-12-07 23:00
6    2021-12-05 05:00
Name: record_date, dtype: object
```

We are going to convert the dates from *object* to *datetime*, specifying the format we want:

```
data_dt['record_date'] = pd.to_datetime(data_dt['record_date'], format = '%Y-%m-%d %H:%M', errors='coerce')
```

```
assert data_dt['record_date'].isnull().sum() == 0, 'missing record date'
```

```
data_dt['record_date'].head()
```

```
2   2021-03-12 15:00:00
3   2021-09-29 09:00:00
4   2021-06-13 11:00:00
5   2021-12-07 23:00:00
6   2021-12-05 05:00:00
Name: record_date, dtype: datetime64[ns]
```

Feature Engineering

We can extract parts of the date and create new columns with that:

```
data_dt['record_date_year'] = data_dt['record_date'].dt.year
data_dt['record_date_month'] = data_dt['record_date'].dt.month
data_dt['record_date_day'] = data_dt['record_date'].dt.day
data_dt['record_date_hour'] = data_dt['record_date'].dt.hour
data_dt['record_date_minute'] = data_dt['record_date'].dt.minute
```

```
data_dt.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents	record_date_year	record_date_month	record_date_day	record_date_hour	record_date_minute
2	1	69	2021-03-12 15:00:00	1	14.0	1025.0	64.0	0.0	1	1	2021	3	12	15	0
3	2	2297	2021-09-29 09:00:00	1	15.0	1028.0	75.0	1.0	1	2	2021	9	29	9	0
4	1	0	2021-06-13 11:00:00	1	27.0	1020.0	52.0	1.0	1	3	2021	6	13	11	0
5	1	0	2021-12-07 23:00:00	2	9.0	1015.0	94.0	0.0	1	4	2021	12	7	23	0
6	1	0	2021-12-05 05:00:00	2	8.0	1026.0	87.0	1.0	1	1	2021	12	5	5	0

Feature Engineering


```
data_dt.nunique()
```

```
magnitude_of_delay        3
delay_in_seconds       1167
record_date            2972
luminosity                3
avg_temperature          34
avg_atm_pressure         34
avg_humidity             80
avg_wind_speed           11
avg_rain                  4
incidents                 4
record_date_year          1
record_date_month        11
record_date_day          31
record_date_hour         24
record_date_minute        1
dtype: int64
```

Since the year and the minute have only one value, we will drop it.

```
data_dt.drop('record_date_year',axis=1,inplace=True)
data_dt.drop('record_date_minute',axis=1,inplace=True)
data_dt.drop('record_date',axis=1,inplace=True)
data_dt.dropna(inplace=True)
```

Feature Engineering

Other functions to deal with dates

```
data_dt2 = data.copy()
```

```
data_dt2['record_date'] = pd.to_datetime(data_dt2['record_date'])
data_dt2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2972 entries, 2 to 4995
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   magnitude_of_delay  2972 non-null   int64
 1   delay_in_seconds    2972 non-null   int64
 2   record_date         2972 non-null   datetime64[ns]
 3   luminosity          2972 non-null   int64
 4   avg_temperature     2972 non-null   float64
 5   avg_atm_pressure    2972 non-null   float64
 6   avg_humidity        2972 non-null   float64
 7   avg_wind_speed      2972 non-null   float64
 8   avg_rain            2972 non-null   int64
 9   incidents           2972 non-null   int64
dtypes: datetime64[ns](1), float64(4), int64(5)
memory usage: 319.9 KB
```

```
data_dt2['record_date'].head()
```

```
2   2021-03-12 15:00:00
3   2021-09-29 09:00:00
4   2021-06-13 11:00:00
5   2021-12-07 23:00:00
6   2021-12-05 05:00:00
Name: record_date, dtype: datetime64[ns]
```

Feature Engineering

We can use *datetime.today* and fetch the actual date.

```
import datetime

today = datetime.datetime.today()

today
```

```
datetime.datetime(2023, 10, 24, 15, 55, 29, 768074)
```

It can be measured the time elapsed between the dates on the dataset and today.

```
today - data_dt2['record_date']
```

```
2       956 days 00:55:29.768074
3       755 days 06:55:29.768074
4       863 days 04:55:29.768074
5       685 days 16:55:29.768074
6       688 days 10:55:29.768074
                  ...
4991    759 days 21:55:29.768074
4992    898 days 04:55:29.768074
4993    852 days 23:55:29.768074
4994    852 days 17:55:29.768074
4995    924 days 15:55:29.768074
Name: record_date, Length: 2972, dtype: timedelta64[ns]
```

```
(today - data_dt2['record_date']).dt.days
```

```
2       956
3       755
4       863
5       685
6       688
       ...
4991    759
4992    898
4993    852
4994    852
4995    924
Name: record_date, Length: 2972, dtype: int64
```

Feature Engineering

And we can also separate each component of the date by day, month, hour, time, etc.

```
data_dt2['day'] = data_dt2['record_date'].dt.day
data_dt2['month'] = data_dt2['record_date'].dt.month
data_dt2['hour'] = data_dt2['record_date'].dt.hour
data_dt2['time'] = data_dt2['record_date'].dt.time
data_dt2.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents	day	month	hour	time
2	1	69	2021-03-12 15:00:00	1	14.0	1025.0	64.0	0.0	1	1	12	3	15	15:00:00
3	2	2297	2021-09-29 09:00:00	1	15.0	1028.0	75.0	1.0	1	2	29	9	9	09:00:00
4	1	0	2021-06-13 11:00:00	1	27.0	1020.0	52.0	1.0	1	3	13	6	11	11:00:00
5	1	0	2021-12-07 23:00:00	2	9.0	1015.0	94.0	0.0	1	4	7	12	23	23:00:00
6	1	0	2021-12-05 05:00:00	2	8.0	1026.0	87.0	1.0	1	1	5	12	5	05:00:00

Feature Engineering

Now we need to choose how to deal with the *record_date*.

```
data['record_date'] = pd.to_datetime(data['record_date'], format = '%Y-%m-%d %H:%M', errors='coerce')
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2972 entries, 2 to 4995
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   magnitude_of_delay 2972 non-null   int64
 1   delay_in_seconds   2972 non-null   int64
 2   record_date        2972 non-null   datetime64[ns]
 3   luminosity         2972 non-null   int64
 4   avg_temperature    2972 non-null   float64
 5   avg_atm_pressure   2972 non-null   float64
 6   avg_humidity       2972 non-null   float64
 7   avg_wind_speed     2972 non-null   float64
 8   avg_rain           2972 non-null   int64
 9   incidents          2972 non-null   int64
dtypes: datetime64[ns](1), float64(4), int64(5)
memory usage: 319.9 KB
```

There are other features that need to be worked on, but it's up to you now!

Feature Engineering

Exploratory Data Analysis

Time to put your data viz skills to the test! Try to recreate the following plots, make sure to import the libraries you'll need!

```
data.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents
2	1	69	2021-03-12 15:00:00	1	14.0	1025.0	64.0	0.0	1	1
3	2	2297	2021-09-29 09:00:00	1	15.0	1028.0	75.0	1.0	1	2
4	1	0	2021-06-13 11:00:00	1	27.0	1020.0	52.0	1.0	1	3
5	1	0	2021-12-07 23:00:00	2	9.0	1015.0	94.0	0.0	1	4
6	1	0	2021-12-05 05:00:00	2	8.0	1026.0	87.0	1.0	1	1

Feature Engineering

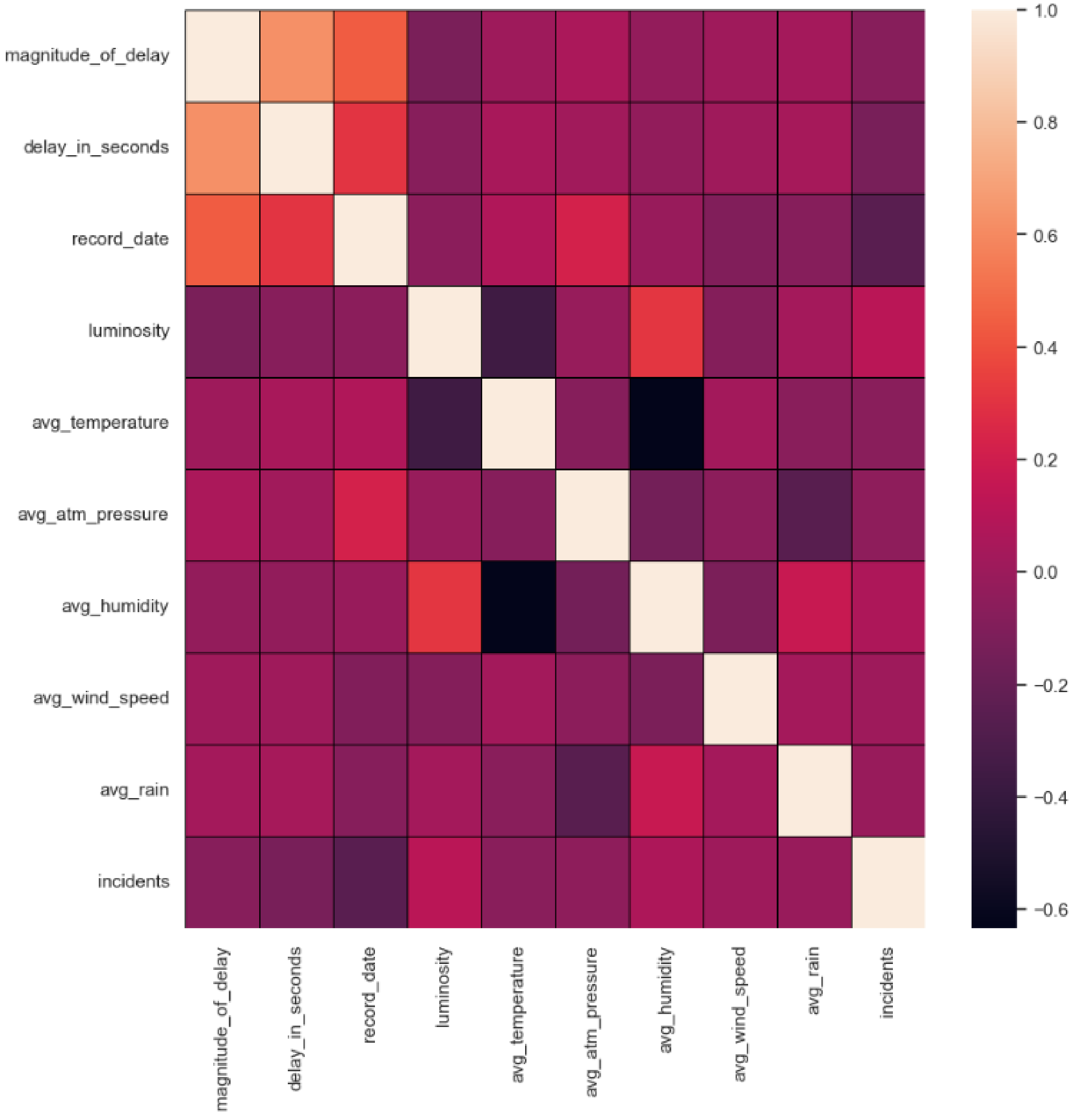
Let's analyze through a heatmap

```
fig = plt.figure( figsize = (10,10))
incidents_corr = data.corr( method = 'pearson')
sns.heatmap(incidents_corr, linecolor='black', linewidths=0.5)
```

We can see that there is a relation between

- *magnitude_of_delay* and *delay_in_seconds*
- *magnitude_of_delay* and *record_date*
- *avg_humidity* and *luminosity*

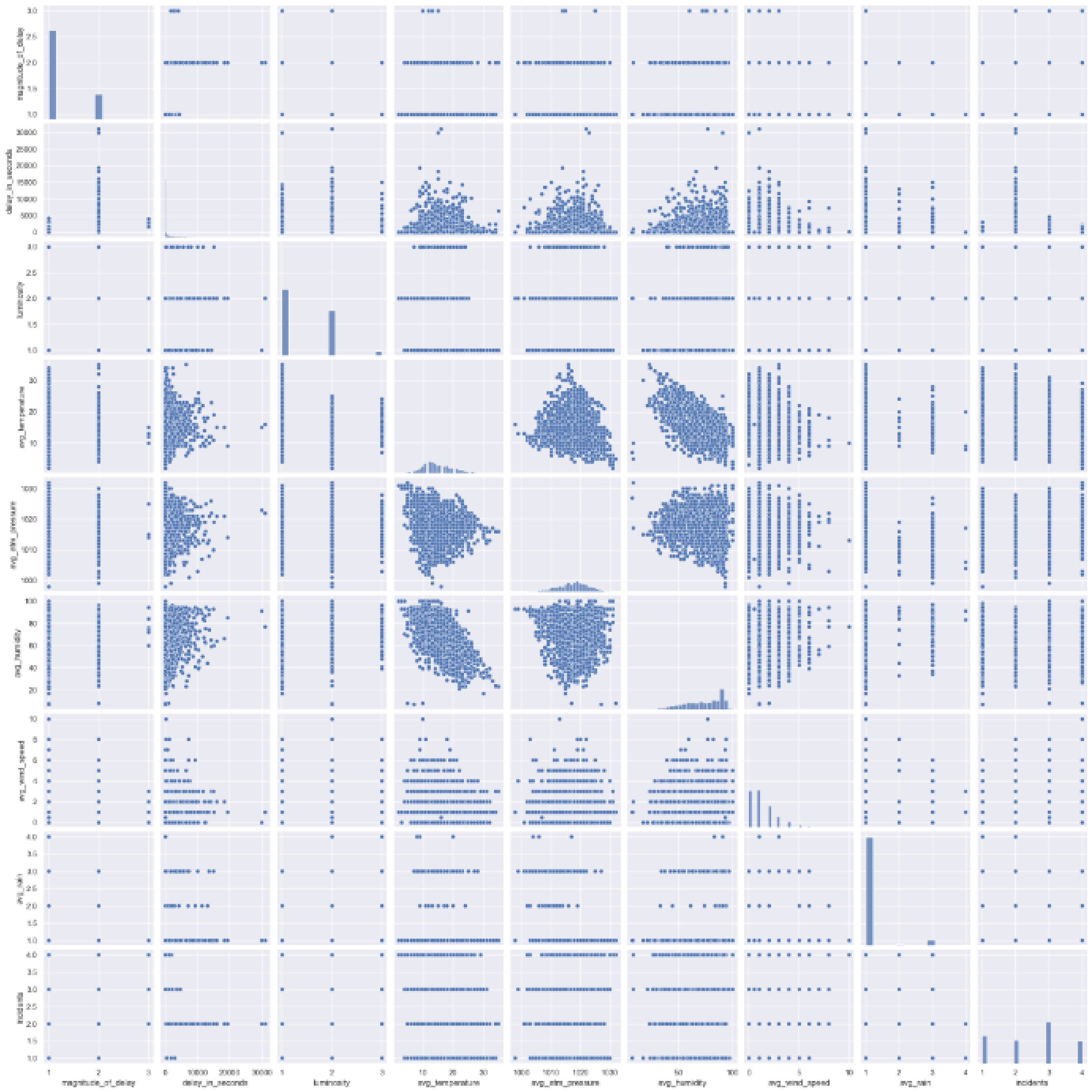
<Axes: >



Feature Engineering


```
sns.pairplot(data)
```

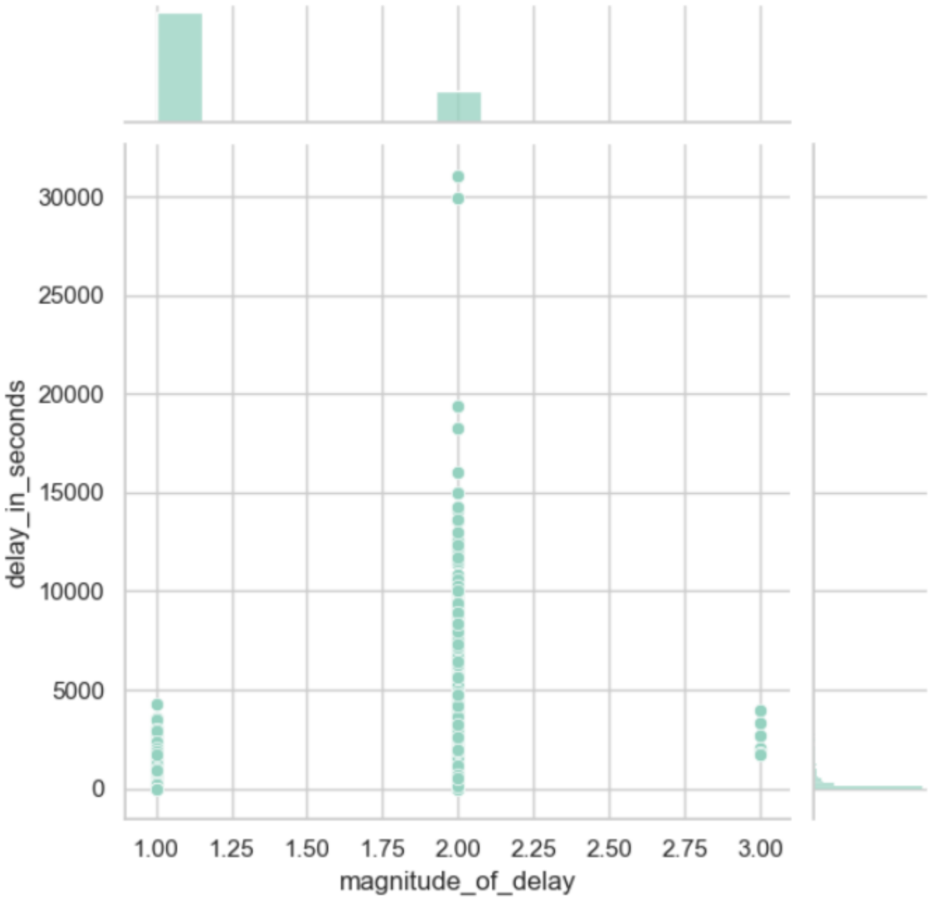
It's hard to analyze the relation of all features. Let's create jointplots between the features with notice a relationship.



Feature Engineering

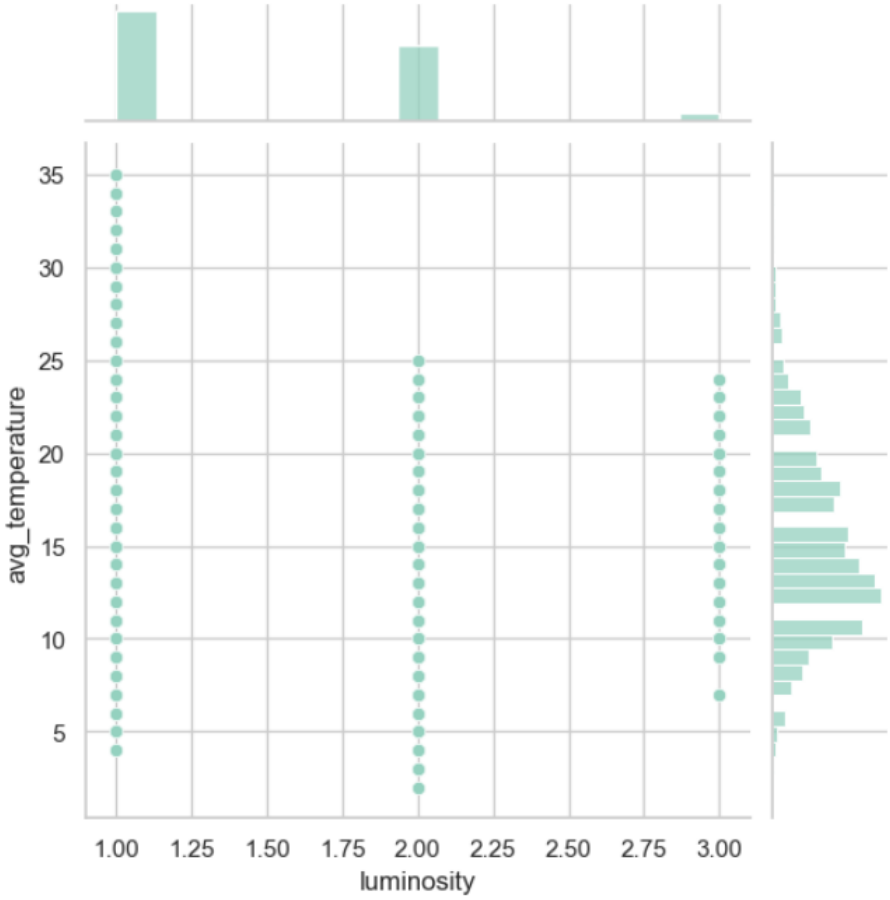
Jointplot of *Magnitude_of_delay vs. Delay_in_seconds*

```
sns.set_palette("GnBu_d")
sns.set_style('whitegrid')
sns.jointplot(x='magnitude_of_delay',y='delay_in_seconds',data=data)
```



Jointplot *Luminosity vs. Avg_temperature*

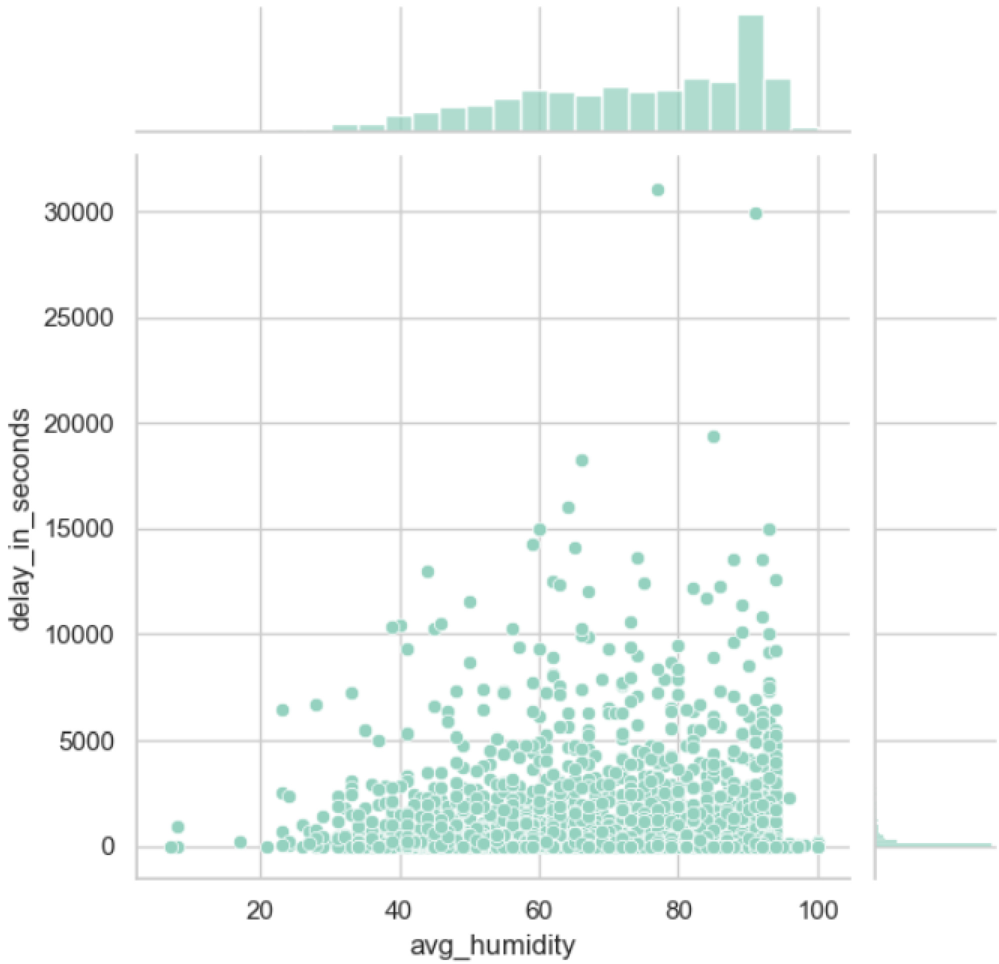
```
sns.jointplot(x='luminosity',y='avg_temperature',data=data)
```



Feature Engineering

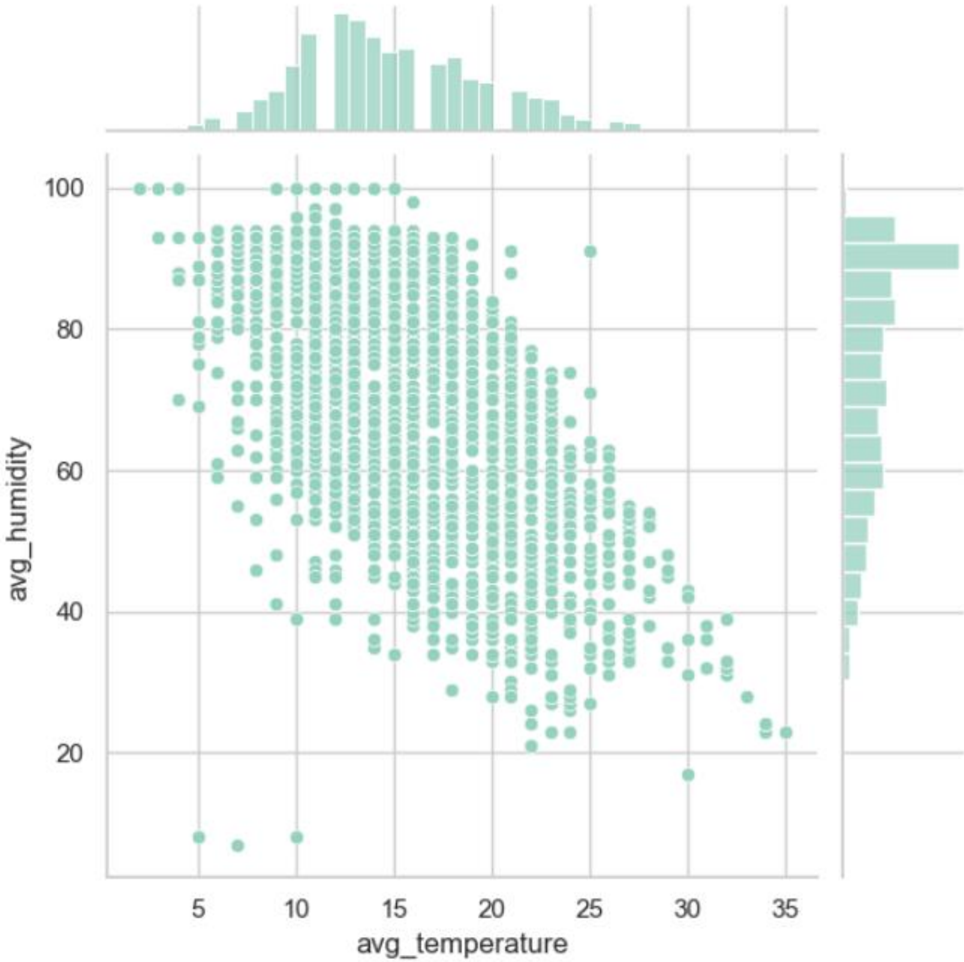
Jointplot *Avg_humidity vs. Delay_in_seconds*

```
sns.jointplot(x='avg_humidity',y='delay_in_seconds',data=data)
```



Jointplot *Avg_temperature vs. Avg_humidity*

```
sns.jointplot(x='avg_temperature',y='avg_humidity',data=data)
```

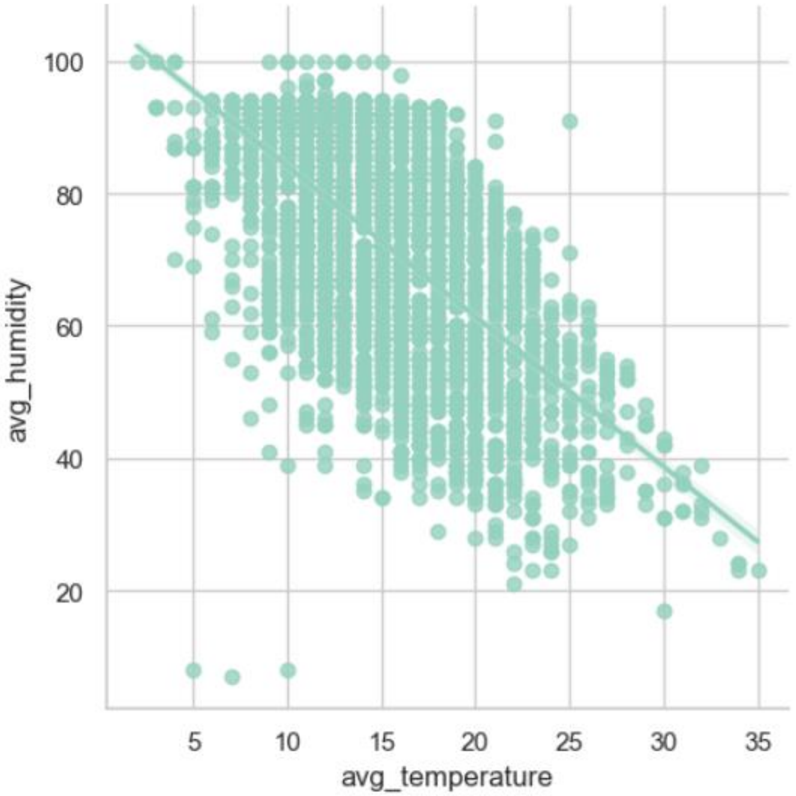


Feature Engineering

It seems there are a relation between *Avg_temperature* and *Avg_humidity*. Let´s create a lmplot

Avg_temperature vs. Avg_humidity

```
sns.lmplot(x='avg_temperature',y='avg_humidity',data=data)
```




```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2972 entries, 2 to 4995
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   magnitude_of_delay  2972 non-null   int64
 1   delay_in_seconds    2972 non-null   int64
 2   record_date         2972 non-null   datetime64[ns]
 3   luminosity          2972 non-null   int64
 4   avg_temperature     2972 non-null   float64
 5   avg_atm_pressure    2972 non-null   float64
 6   avg_humidity        2972 non-null   float64
 7   avg_wind_speed      2972 non-null   float64
 8   avg_rain            2972 non-null   int64
 9   incidents           2972 non-null   int64
dtypes: datetime64[ns](1), float64(4), int64(5)
memory usage: 319.9 KB
```

```
data.head()
```

	magnitude_of_delay	delay_in_seconds	record_date	luminosity	avg_temperature	avg_atm_pressure	avg_humidity	avg_wind_speed	avg_rain	incidents
2	1	69	2021-03-12 15:00:00	1	14.0	1025.0	64.0	0.0	1	1
3	2	2297	2021-09-29 09:00:00	1	15.0	1028.0	75.0	1.0	1	2
4	1	0	2021-06-13 11:00:00	1	27.0	1020.0	52.0	1.0	1	3
5	1	0	2021-12-07 23:00:00	2	9.0	1015.0	94.0	0.0	1	4
6	1	0	2021-12-05 05:00:00	2	8.0	1026.0	87.0	1.0	1	1