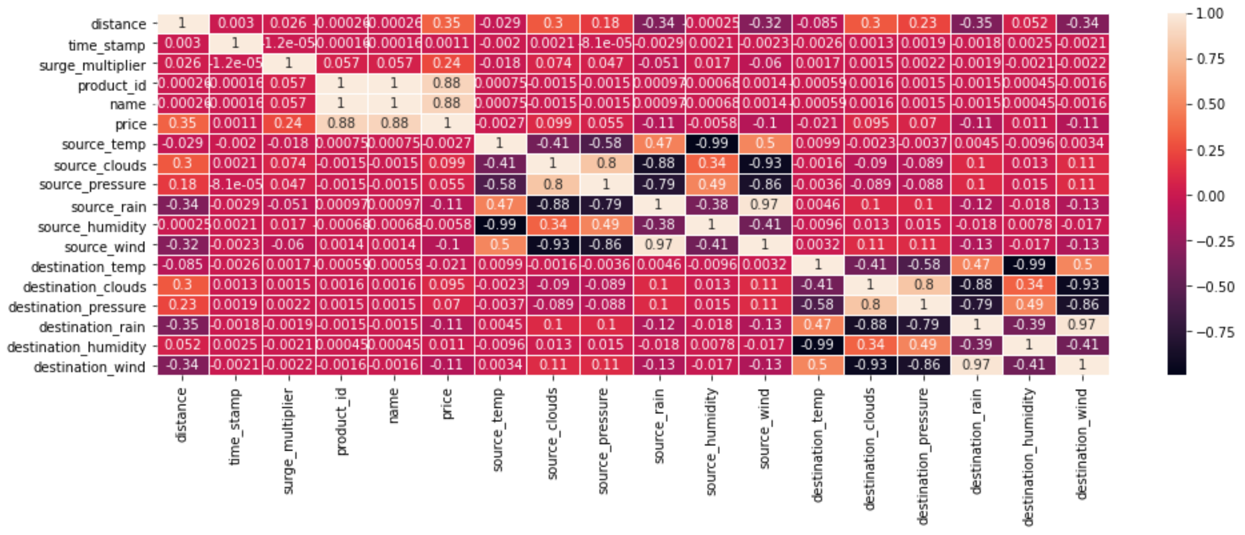
Data description

Since we are dealing with two different datasets we have had to find a connection between two of them, which we decided to connect them based on the source/destination locations
So we merged the weather dataset two times for the source and destination by grouping the data by location then rename the columns to their respected names (i.e : location -> destination location -> source)

preprocessing techniques

Remove dublications

We found out by analyzing the data that the two columns `product_id` and `name` Are connected with each other which indicates that both are highly correlated so we opt to remove `product_id` since it’s described as a duplication



from the figure we can see that there are multiple features that are considered highly correlated with each other that we can drop which are:

- source_humidity -> source_temp
- source_wind -> source_cloud
- destination_humidity -> destination_temp
- destination_wind -> destination_cloud
- Product_id -> name

remove outliers

We used Zscore function to remove outliers from columns :

- 'distance',
- 'source_clouds',
- 'source_temp',
- 'source_pressure',
- 'source_rain',
- 'destination_clouds',
- 'destination_temp',
- 'destination_pressure',
- 'destination_rain'

Data Encoding

Using mean Encoding to encode data since it has great balance between efficiency and model complexity but the trade off was that we were forced to do the outlier removal before the encoding to make sure that the model will not overfitting

Feature selection

By studying the correlation between the data and the label we have found that price label doesn't have great correlation with the other so we choose based on correlation bigger than 0.2 Which lead us to use

- `Distance`
- `name`

Feature Scaling

We used MinMaxScaler() to scale the data

Model Training

First we split the data into 20% test and 80% train then we applied `two models with and without validation techniques:`

- ## linear_model.LinearRegression()

  ### Without cross validation

  ```
  At degree = 8
  Train set size: 306192
  ```

```
Test set size: 204129
Train subset (MSE) for degree 8:  2.0420390066419776
Test subset (MSE) for degree 8:  2.0418024342617693
Training time: 1.6s
```

With cross validation cv = 9 and degree 8

model cross validation score is 2.042685005167811
Training time: 7s

- linear_model.Ridge()

Without cross validation

```
At degree = 12
Train set size: 219064
Test set size: 54767
Train subset (MSE) for degree 12:  2.72520925571384
Test subset (MSE) for degree 12:  2.7429110428911168
Training time: 0.9s
```

With cross validation cv = 9 and degree 8

model cross validation score is 2.088146425934931
Training time: 7s

Conclusion

We found out that rain is import for the prediction despite having 82% null values
So what we did was creating a secondary model to predict the value of the rain
before working on the main dataset
And it helped with the MSE:
When dropping the rain column the MSE was around : 2.9
When using it the MSE was : 2.088
Also another problem that we ran into was with the encoding method we used which is mean encoding: despite being very efficient and got us really good score but we a had a serious potential of data leaking so our fix for the problem was to use k-fold validation but not to valid the result instead we used it to use different points each time we ran the mean encoding code to prevent the data leaking issue, later we foud out that the mean encoding function in the sklearn api do that exactly so we used it in the end, but the trade-off was that our model prediction would differ with each run