RAMP PROJECT – AIR PASSENGER DATA

FINAL PRESENTATION | PYTHON FOR DATA SCIENCE | DECEMBER 2020

Table of contents

1. Exploratory Data Analysis

Overview of the variables

Dataset statistics

Number of variables	6
Number of observations	8902
Missing cells	0

Variable types

CAT	3
NUM	3

Qualitative Variables

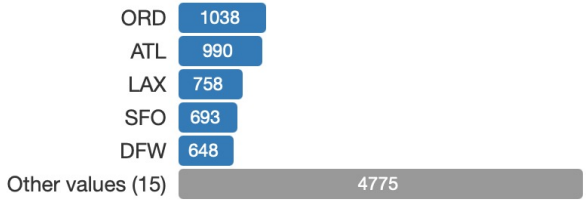
DateOfDeparture
Categorical

`HIGH CARDINALITY`

Distinct	552
Distinct (%)	6.2%
Missing	0
Missing (%)	0.0%
Memory size	69.5 KiB

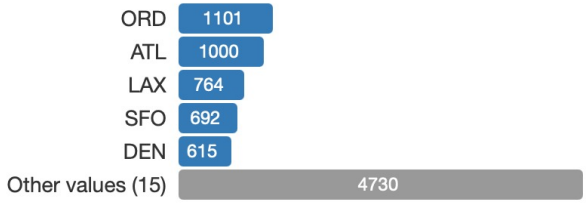
Departure
Categorical

Distinct	20
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	69.5 KiB



Arrival
Categorical

Distinct	20
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	69.5 KiB



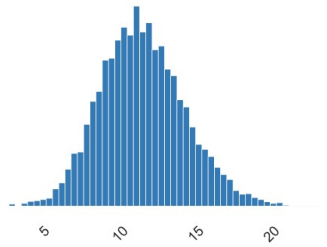
Quantitative variables

WeeksToDeparture

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	1867
Distinct (%)	21.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	11.44646878
Minimum	2.625
Maximum	23.16326531
Zeros	0
Zeros (%)	0.0%
Memory size	69.5 KiB

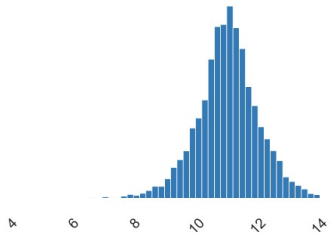


log_PAX

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	1122
Distinct (%)	12.6%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	10.99904767
Minimum	3.87810806
Maximum	14.00779141
Zeros	0
Zeros (%)	0.0%
Memory size	69.5 KiB

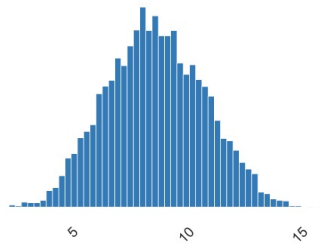


std_wtd

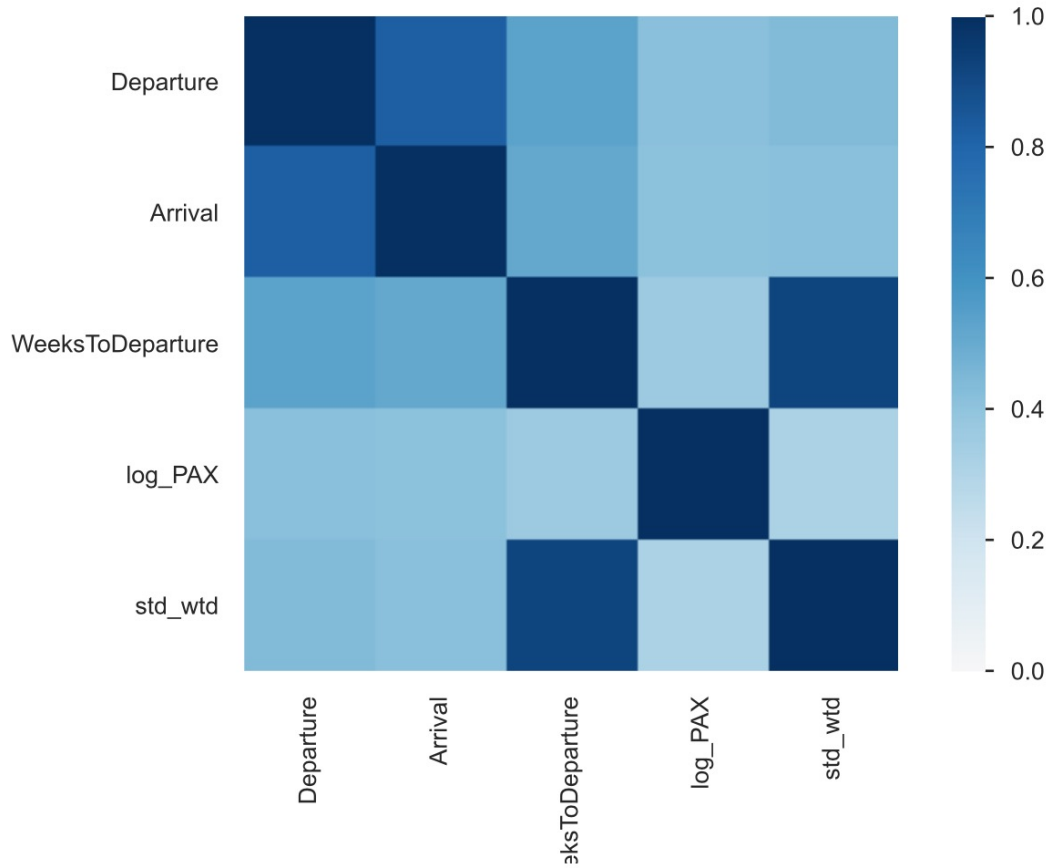
Real number ($\mathbb{R}_{\geq 0}$)

Distinct	7723
Distinct (%)	86.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	8.617772555
Minimum	2.160246899
Maximum	15.8622158
Zeros	0
Zeros (%)	0.0%
Memory size	69.5 KiB



Correlation table

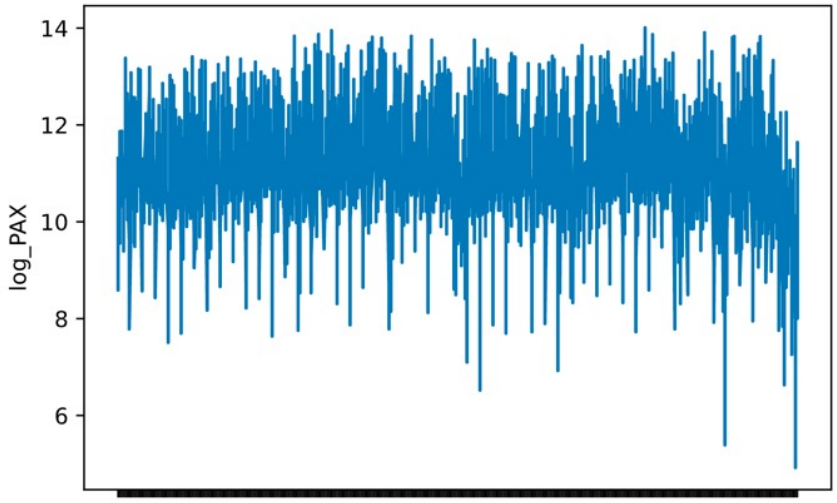


First rows

	DateOfDeparture	Departure	Arrival	WeeksToDeparture	log_PAX	std_wtd
0	2012-06-19	ORD	DFW	12.875000	12.331296	9.812647
1	2012-09-10	LAS	DEN	14.285714	10.775182	9.466734
2	2012-10-05	DEN	LAX	10.863636	11.083177	9.035883
3	2011-10-09	ATL	ORD	11.480000	11.169268	7.990202
4	2012-02-21	DEN	SFO	11.450000	11.269364	9.517159
5	2013-01-22	ATL	MCO	10.363636	12.073649	8.232025
6	2011-10-20	SFO	LAS	15.266667	11.173936	9.808277
7	2012-01-28	EWR	ORD	8.588235	9.599952	6.165010
8	2012-05-27	ATL	CLT	10.238095	9.175645	6.609877
9	2013-02-22	ATL	DEN	8.294118	10.734320	5.542616

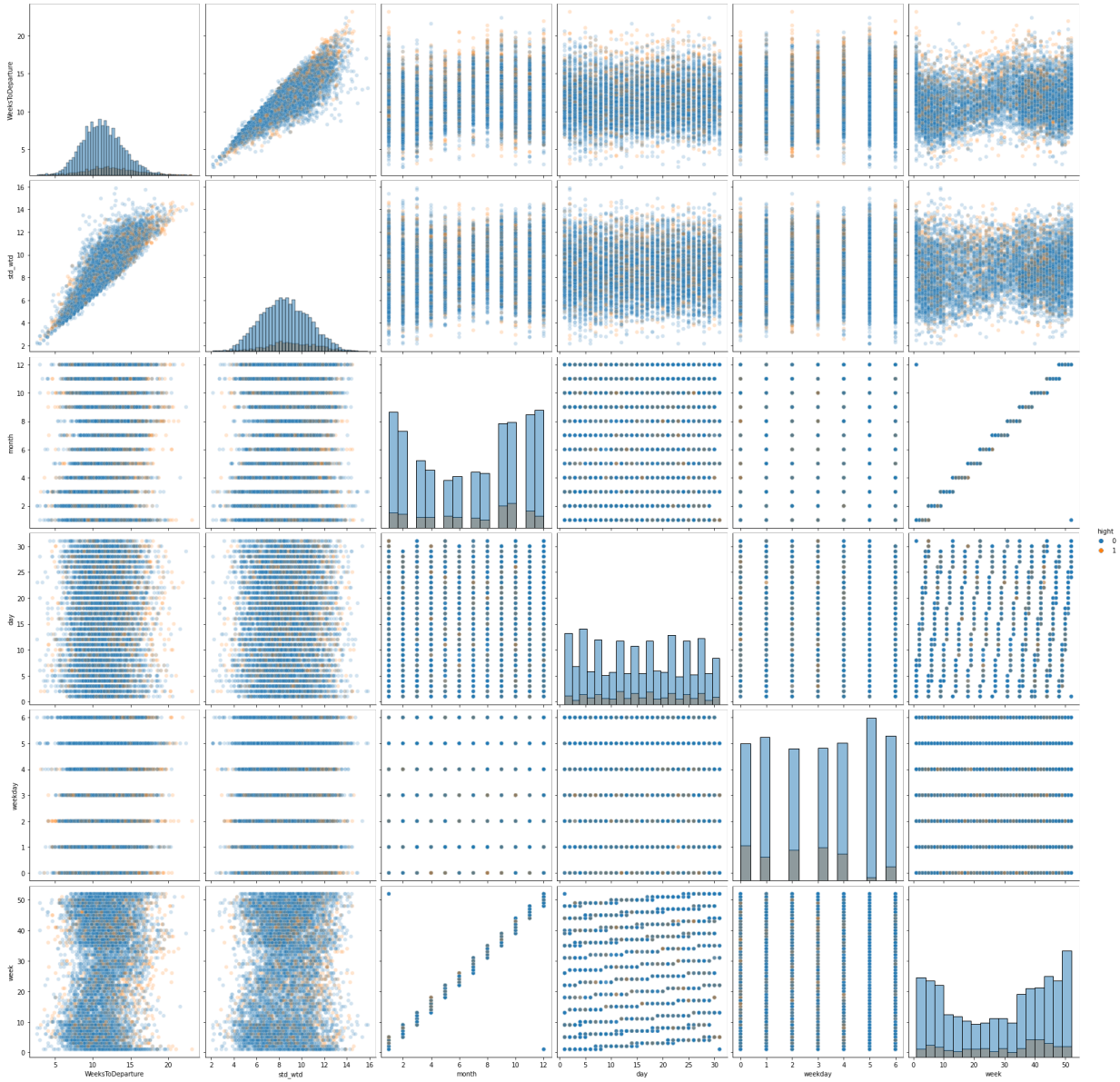
Further exploration of high log_PAX values

month	1	2	3	4	5	6	7	8	9	10	11	12
hight												
0	88.480636	87.558685	86.212625	84.659091	79.520697	82.401656	85.009862	86.983471	82.25641	81.15653	86.913087	90.709291
1	11.519364	12.441315	13.787375	15.340909	20.479303	17.598344	14.990138	13.016529	17.74359	18.84347	13.086913	9.290709



weekday	0	1	2	3	4	5	6
hight							
0	79.562594	85.680934	81.28	79.890454	83.491311	97.887324	91.140279
1	20.437406	14.319066	18.72	20.109546	16.508689	2.112676	8.859721

Pairplots



2. Adding dataset

Adding the weather data

	DateOfDeparture	Arrival	Max TemperatureC	Mean TemperatureC	Min TemperatureC	Dew PointC	MeanDew PointC	Min DewpointC	Max Humidity	Mean Humidity	Min Humidity	Max Sea Level PressurehPa
0	2011-09-01	ATL	35	29	24	21	18	14	79	56	32	1022
1	2011-09-02	ATL	36	29	22	17	15	14	61	46	30	1019
2	2011-09-03	ATL	35	29	23	17	16	14	64	47	30	1015
3	2011-09-04	ATL	27	24	22	22	19	16	93	72	51	1014
4	2011-09-05	ATL	26	24	22	23	22	20	94	91	87	1010
...	...	...	...	...	...	...	...	...	...	...	...	...
11035	2013-03-01	LGA	7	5	3	-1	-3	-4	76	63	49	1008
11036	2013-03-02	LGA	4	2	0	-2	-5	-6	82	65	48	1008
11037	2013-03-03	LGA	4	2	-1	-5	-8	-9	69	55	40	1008
11038	2013-03-04	LGA	5	2	-2	-7	-8	-9	63	54	44	1012
11039	2013-03-05	LGA	9	5	1	-3	-5	-7	61	49	37	1016

Splitting the date of departure

	Departure	Arrival	WeeksToDeparture	log_PAX	std_wtd	year	month	day	weekday	week	n_days
0	ORD	DFW	12.875000	12.331296	9.812647	2012	6	19	1	25	15510
1	LAS	DEN	14.285714	10.775182	9.466734	2012	9	10	0	37	15593
2	DEN	LAX	10.863636	11.083177	9.035883	2012	10	5	4	40	15618
3	ATL	ORD	11.480000	11.169268	7.990202	2011	10	9	6	40	15256
4	DEN	SFO	11.450000	11.269364	9.517159	2012	2	21	1	8	15391
...	...	...	...	...	...	...	...	...	...	...	...
8897	DTW	ATL	9.263158	10.427055	7.316967	2011	10	2	6	39	15249
8898	DFW	ORD	12.772727	12.201552	10.641034	2012	9	25	1	39	15608
8899	SFO	LAS	11.047619	10.508746	7.908705	2012	1	19	3	3	15358
8900	ORD	PHL	6.076923	10.174042	4.030334	2013	2	3	6	5	15739
8901	DTW	ATL	9.526316	9.202674	6.167733	2011	11	26	5	47	15304

Adding the holidays data

	DateOfDeparture	Departure	Arrival	WeeksToDeparture	log_PAX	std_wtd	Holiday
24	2012-11-12	EWR	BOS	11.000000	11.740518	8.711759	Veterans Day
68	2012-12-25	DEN	LAX	15.034483	10.391932	11.890821	Christmas Day
77	2013-01-21	DEN	ORD	9.631579	10.935530	7.158604	Birthday of Martin Luther King, Jr.
119	2011-11-24	DEN	ORD	10.176471	8.285437	7.307832	Thanksgiving Day
161	2013-01-01	LAX	BOS	13.666667	9.814790	8.948571	New Year's Day
...	...	...	...	...	...	...	...
8788	2012-10-08	MSP	DEN	9.631579	11.003166	6.701663	Columbus Day
8802	2012-01-02	EWR	SFO	15.037037	11.269364	11.686552	New Year's Day
8819	2012-05-28	SFO	JFK	16.096774	12.293312	11.249755	Memorial Day
8843	2012-12-25	SFO	LAX	17.705882	11.048593	11.645078	Christmas Day
8857	2011-10-10	PHX	ORD	8.888889	10.908849	6.182412	Columbus Day

We tried different approach for the holidays

Key highlights

1 Chritmas	• The weekend of the 51th week of the year • The 52th and 1st week of the year
2 Labor weekend	• The weekend of the 35th week of the year • The Monday of the 36th week of the year
3 Memorial weekend	• The weekend of the 21th week of the year • The Monday of the 22th week of the year
4 4th of July	• The weekend of the 26th week of the year • The 27th week of the year
5 Thanksgiving	• The weekend of the 46th week of the year • The 47th week of the year
6 Summer Friday	• All Friday between the 22th and 35th week of the year

Adding the holidays data

Keeping only a binary format for the holidays

	Departure	Arrival	WeeksToDeparture	log_PAX	month	day	weekday	week	n_days	holiday
0	ORD	DFW	12.875000	12.331296	6	19	1	25	15510	0
1	LAS	DEN	14.285714	10.775182	9	10	0	37	15593	0
2	DEN	LAX	10.863636	11.083177	10	5	4	40	15618	0
3	ATL	ORD	11.480000	11.169268	10	9	6	40	15256	0
4	DEN	SFO	11.450000	11.269364	2	21	1	8	15391	0
...	...	...	...	...	...	...	...	...	...	...
8897	DTW	ATL	9.263158	10.427055	10	2	6	39	15249	0
8898	DFW	ORD	12.772727	12.201552	9	25	1	39	15608	0
8899	SFO	LAS	11.047619	10.508746	1	19	3	3	15358	0
8900	ORD	PHL	6.076923	10.174042	2	3	6	5	15739	0
8901	DTW	ATL	9.526316	9.202674	11	26	5	47	15304	1

3. Preprocessing

Encoding the data – Qualitative Variables

Label encoding of "Departure" and "Arrival"

	Departure	Arrival	WeeksToDeparture	log_PAX	std_wtd	month	day	weekday	week	n_days	holiday
0	15	4	12.875000	12.331296	9.812647	6	19	1	25	15510	0
1	9	3	14.285714	10.775182	9.466734	9	10	0	37	15593	0
2	3	10	10.863636	11.083177	9.035883	10	5	4	40	15618	0
3	0	15	11.480000	11.169268	7.990202	10	9	6	40	15256	0
4	3	19	11.450000	11.269364	9.517159	2	21	1	8	15391	0
...	...	...	...	...	...	...	...	...	...	...	...
8897	5	0	9.263158	10.427055	7.316967	10	2	6	39	15249	0
8898	4	15	12.772727	12.201552	10.641034	9	25	1	39	15608	0
8899	19	9	11.047619	10.508746	7.908705	1	19	3	3	15358	0
8900	15	16	6.076923	10.174042	4.030334	2	3	6	5	15739	0
8901	5	0	9.526316	9.202674	6.167733	11	26	5	47	15304	1

Encoding the data – Quantitative Variables

Normalization of "WeeksToDeparture" and "std_wtd"

	Departure	Arrival	WeeksToDeparture	log_PAX	std_wtd	month	day	weekday	week	n_days	holiday
0	15	4	0.512573	12.331296	0.558487	6	19	1	25	15510	0
1	9	3	1.018752	10.775182	0.396807	9	10	0	37	15593	0
2	3	10	-0.209127	11.083177	0.195426	10	5	4	40	15618	0
3	0	15	0.012031	11.169268	-0.293328	10	9	6	40	15256	0
4	3	19	0.001267	11.269364	0.420375	2	21	1	8	15391	0
...	...	...	...	...	...	...	...	...	...	...	...
8897	5	0	-0.783396	10.427055	-0.608000	10	2	6	39	15249	0
8898	4	15	0.475876	12.201552	0.945677	9	25	1	39	15608	0
8899	19	9	-0.143112	10.508746	-0.331420	1	19	3	3	15358	0
8900	15	16	-1.926652	10.174042	-2.144180	2	3	6	5	15739	0
8901	5	0	-0.688972	9.202674	-1.145154	11	26	5	47	15304	1

Encoding the data – Quantitative Variables

Kbins discretizer using kmeans on the variables 'WeeksToDeparture' and 'std_wtd'

	Departure	Arrival	WeeksToDeparture	log_PAX	std_wtd	year	month	day	weekday	week	n_days	holiday
0	15	4	5.0	12.331296	5.0	2012	6	19	1	25	15510	0
1	9	3	6.0	10.775182	5.0	2012	9	10	0	37	15593	0
2	3	10	3.0	11.083177	4.0	2012	10	5	4	40	15618	0
3	0	15	4.0	11.169268	3.0	2011	10	9	6	40	15256	0
4	3	19	4.0	11.269364	5.0	2012	2	21	1	8	15391	0
...	...	...	...	...	...	...	...	...	...	...	...	...
8897	5	0	2.0	10.427055	2.0	2011	10	2	6	39	15249	0
8898	4	15	5.0	12.201552	6.0	2012	9	25	1	39	15608	0
8899	19	9	3.0	10.508746	3.0	2012	1	19	3	3	15358	0
8900	15	16	0.0	10.174042	0.0	2013	2	3	6	5	15739	0
8901	5	0	2.0	9.202674	1.0	2011	11	26	5	47	15304	1

4. Machine Learning methods

We tried different Machine Learning Algorithms

Key highlights

1 Linear Regression	• The model didn't show very promising results • Low correlation between the variables 'WeeksToDeparture'and 'std_wtd' and the response variable 'log_pax'
2 LinearGAM	• Can lead to a smoother result than the Linear regression mode • Easy to interpret and has a specific treatment for parametric variables just like in an Anova regression • Promising result, like an 0.92 R-squared value
3 K-Means	• Showed bad results
4 Neural Network	• We used the package 'sklearn.neural_network.MLPRegressor' • Lack of knowledge on this model and the very high standard deviation of the model prediction
5 Decision Tree Regressor	• Simple model leading to quite interesting results • Using the function 'plot_tree', the model can be easily interpreted
6 XGBoost	• The XGB model is a random forest model (multitude of decision trees) boosted by gradient • The most accurate model on our dataset

Linear GAM

$$F(x) = y = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_r(x_r)$$

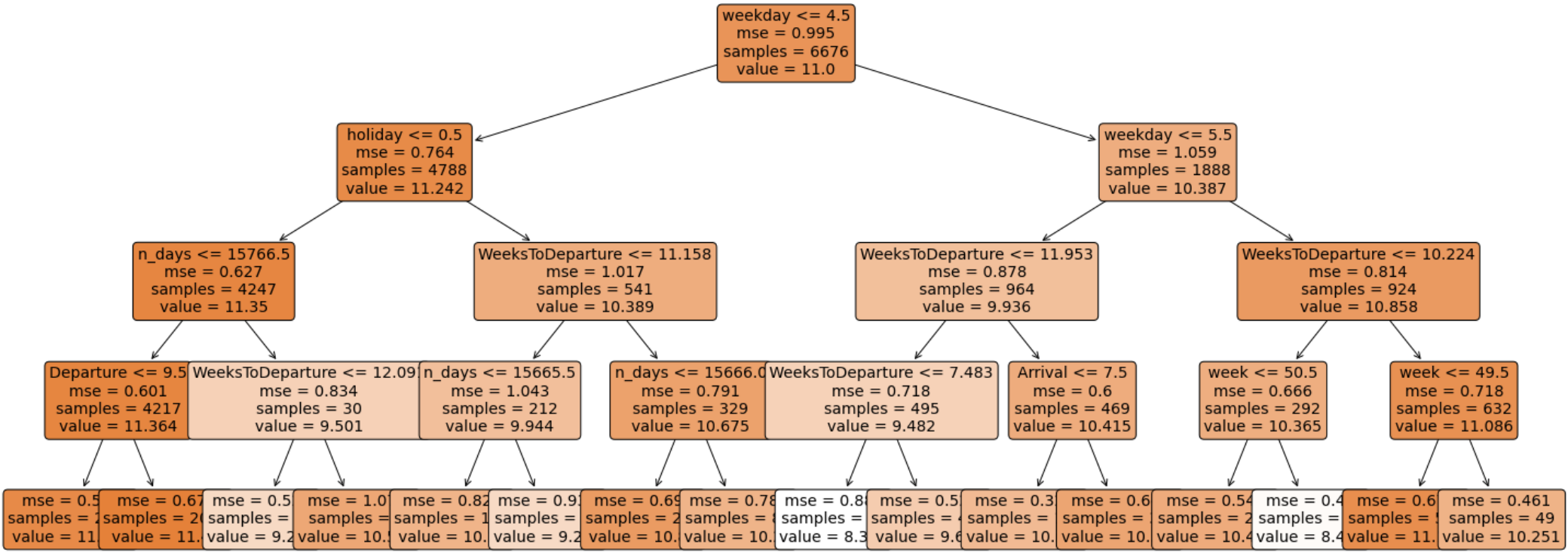
LinearGAM

Distribution:	NormalDist	Effective DoF:	142.2319
Link Function:	IdentityLink	Log Likelihood:	-8483.7372
Number of Samples:	6676	AIC:	17253.9384
		AICc:	17260.2639
		GCV:	0.3608
		Scale:	0.347
		Pseudo R-Squared:	0.9747

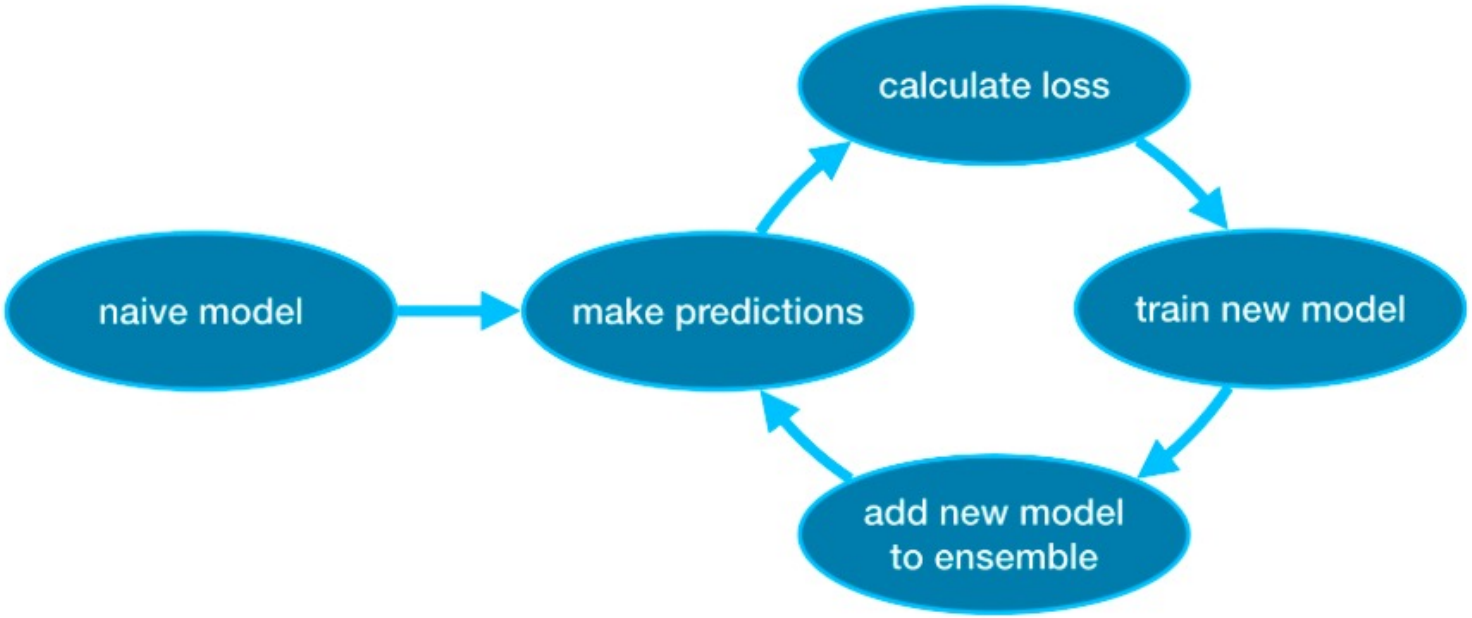
Feature Function	Lambda	Rank	EDoF	P > x	Sig. Code
f(0)	[0.2512]	20	20.0	1.11e-16	***
f(1)	[0.2512]	20	19.0	1.11e-16	***
s(2)	[0.2512]	20	14.3	1.11e-16	***
s(3)	[0.2512]	20	15.0	1.99e-02	*
f(4)	[0.2512]	2	1.0	5.46e-01	
f(5)	[0.2512]	4	0.0	1.11e-16	***
f(6)	[0.2512]	13	11.0	1.11e-16	***
f(7)	[0.2512]	7	6.0	1.11e-16	***
f(8)	[0.2512]	53	50.0	1.11e-16	***
f(9)	[0.2512]	2	1.0	9.52e-06	***
f(10)	[0.2512]	2	1.0	2.50e-05	***
f(11)	[0.2512]	2	1.0	1.11e-16	***
f(12)	[0.2512]	2	1.0	3.01e-01	
f(13)	[0.2512]	2	1.0	6.50e-03	**
f(14)	[0.2512]	2	1.0	4.30e-01	
intercept		1	0.0	1.11e-16	***

Significance codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Decision Tree

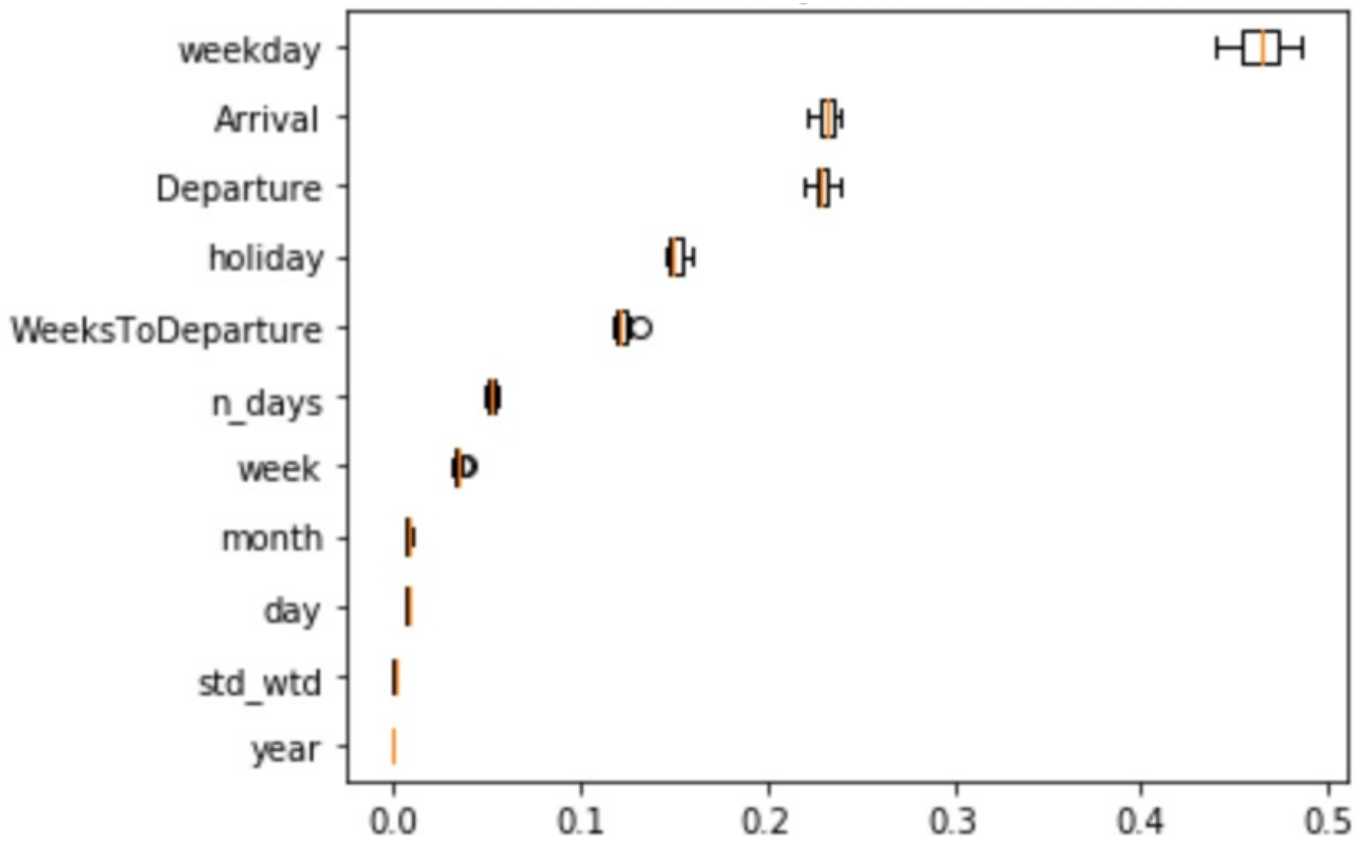


XGB


calculate loss
naive model
make predictions
train new model
add new model
to ensemble

5. Features' selection

Permutation importance on the train set



6. Hyper parameters

Here are the different hyper parameters for our XGBoost model

We used a 'GridSearchCV' algorithm to tune them

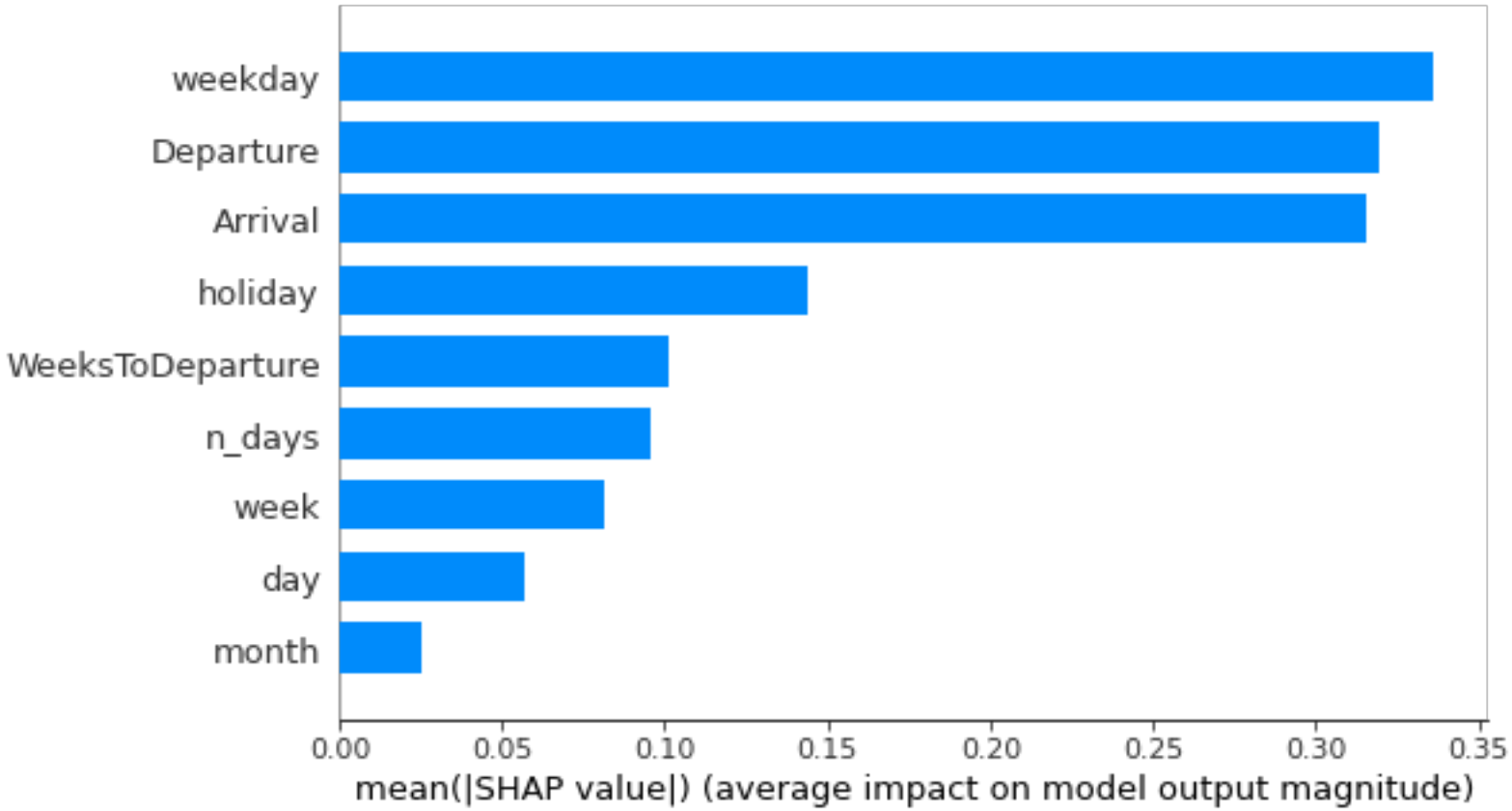
1 | n_estimator	• specifies how many times to go through the cycle. It is equal to the number of models that we include in the set. A value that is too low leads to under-fitting. One that is too high causes overfitting
2 | learning_rate	• the predictions from each model are multiplied by a small number (known as the learning rate), so that each tree we add to our model help us a little less. A small value of learning rate allows the use of larger n_estimators.
3 | subsample	• it is the subsample ratio of the training instances. By reducing it a little we make our model more robust to noise
4 | colampse_ bytree	• it is the subsample ratio of columns when constructing each tree, it will also make our model more robust to noise
5 | max_depth	• this parameter increases the depth of each tree. It will make the model more complex but also more likely to overfit
6 | early_stopping_ rounds	• this parameter also permit to avoid over fitting as it stop the loop if the model does not improve after a few trials. This way we can increase our n_estimators.

GridSearchCV

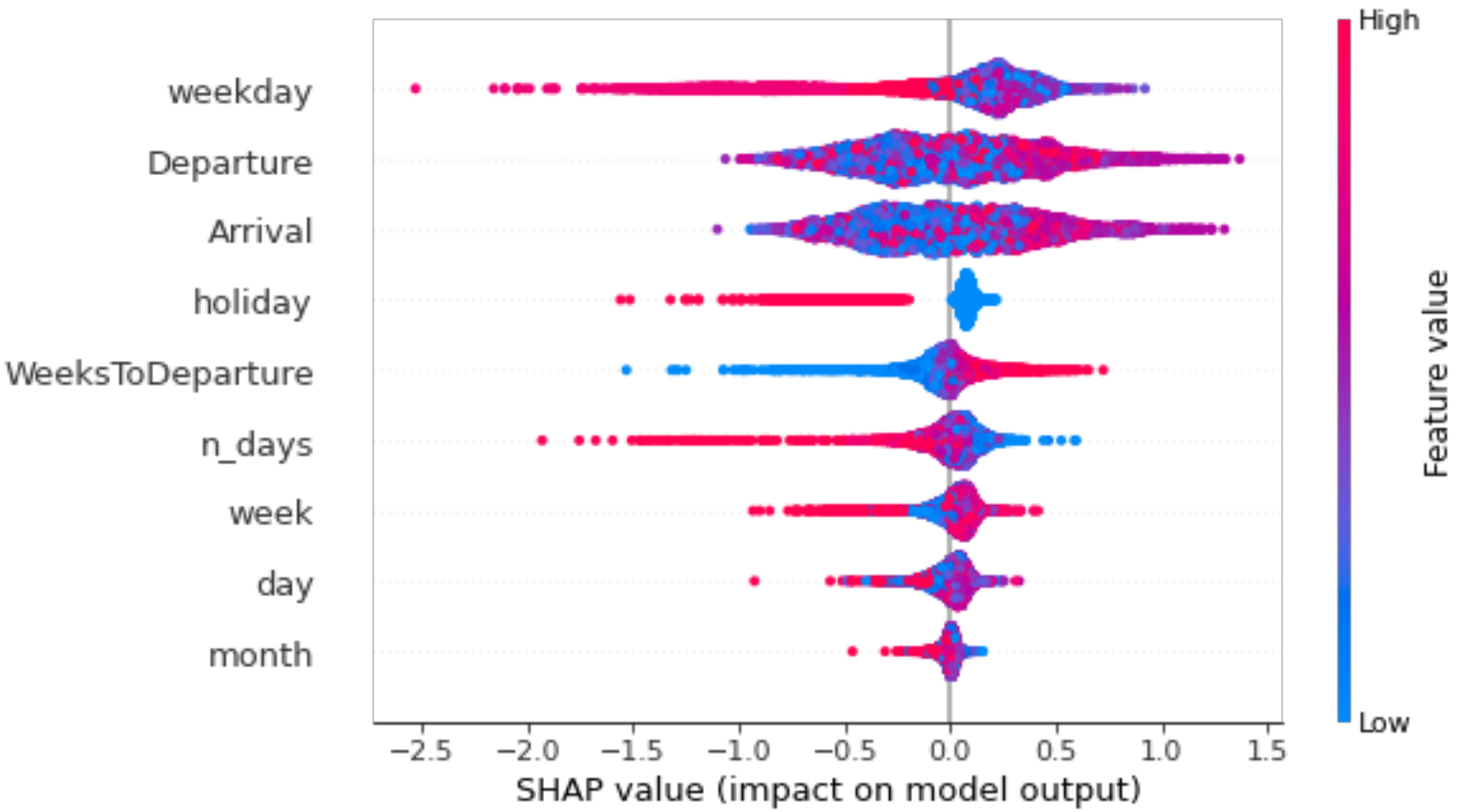
```
Fitting 5 folds for each of 135 candidates, totalling 675 fits
[Parallel(n_jobs=5)]: Using backend LokyBackend with 5 concurrent workers.
[Parallel(n_jobs=5)]: Done  40 tasks      | elapsed:  1.3min
[Parallel(n_jobs=5)]: Done 190 tasks      | elapsed: 12.1min
[Parallel(n_jobs=5)]: Done 440 tasks      | elapsed: 30.1min
[Parallel(n_jobs=5)]: Done 675 out of 675 | elapsed: 45.6min finished
[23:51:39] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:516:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


-0.13221370992646236
{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 10, 'min_child_weight': 5, 'n_estimators': 1000, 'nth
read': 4, 'objective': 'reg:squarederror', 'silent': 1, 'subsample': 0.7}
```

7. Interpretability

Variables importance graph



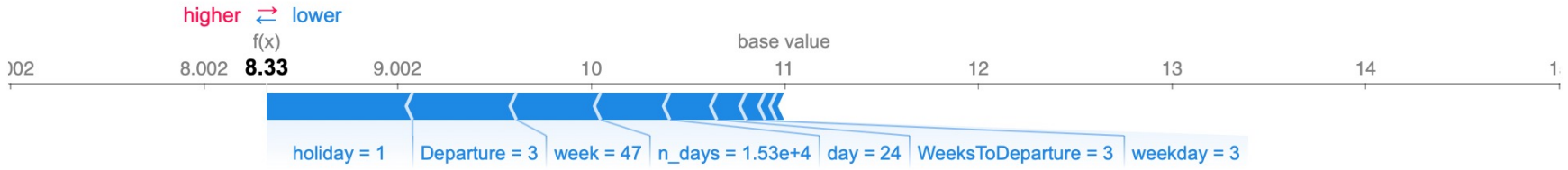
Summary plot



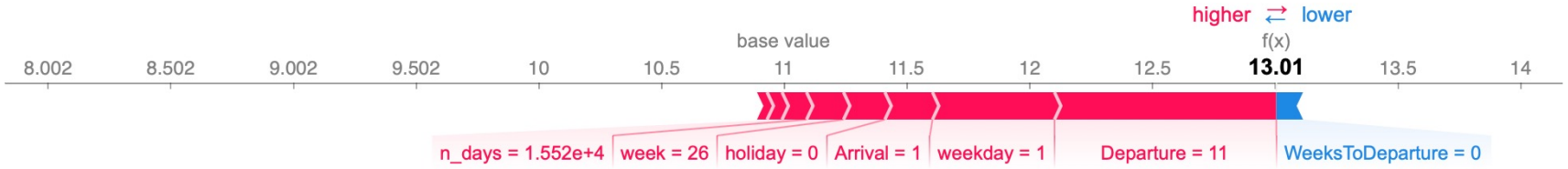
Force plots

For the observations number 119 and 131

```
For the 119 th value the predicted value of y is 8.330065 . Nevertheless, the true value of y is log_PAX    8.28543
7
Name: 119, dtype: float64
```



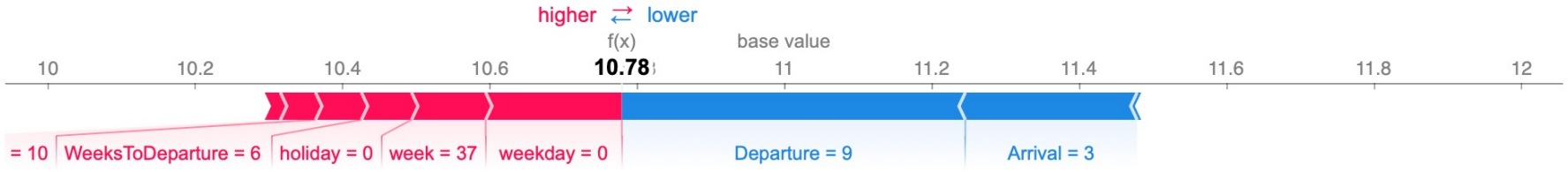

```
For the 131 th value the predicted value of y is 13.009695 . Nevertheless, the true value of y is log_PAX    13.008
978
Name: 131, dtype: float64
```



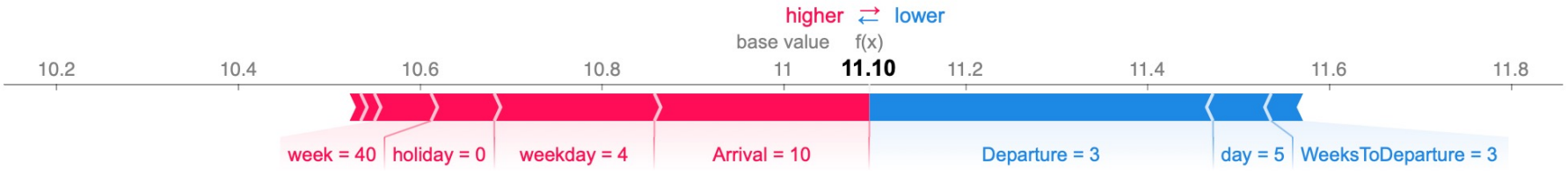
Force plots

For the observations number 1, 2

```
For the 1 th value the predicted value of y is 10.782328 . Nevertheless, the true value of y is log_PAX    10.77518
2
Name: 1, dtype: float64
```

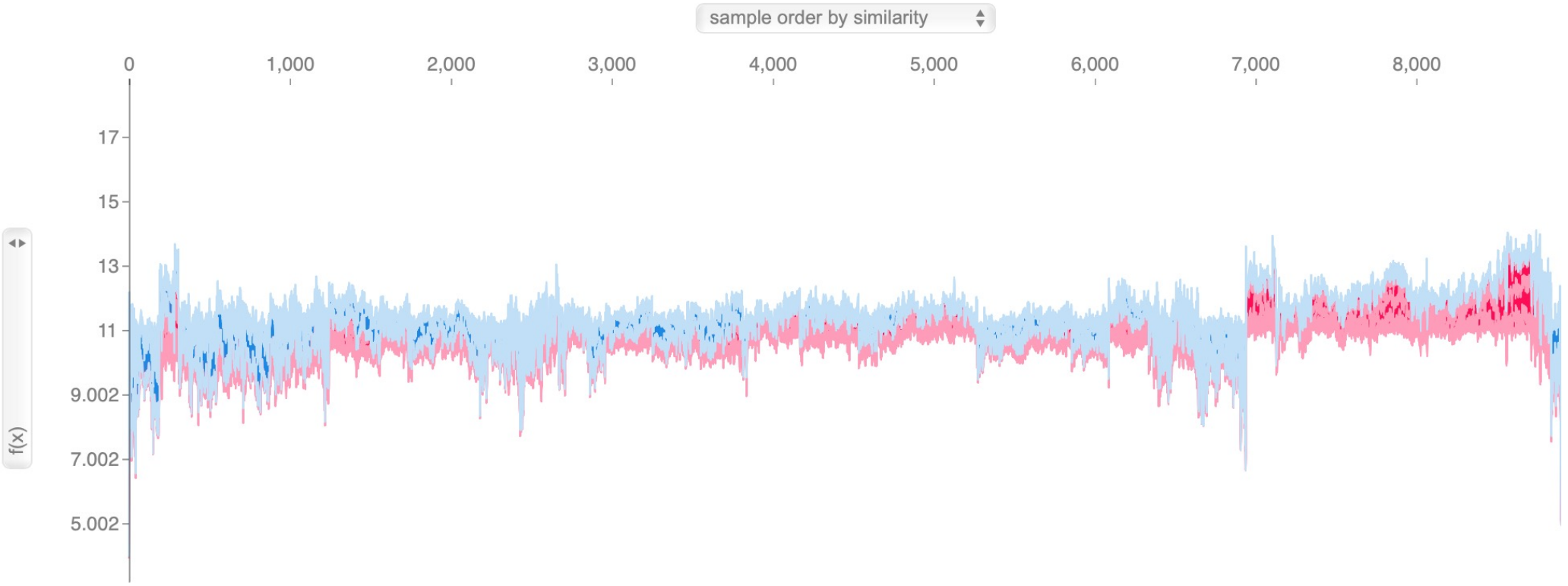



```
For the 2 th value the predicted value of y is 11.097139 . Nevertheless, the true value of y is log_PAX    11.08317
7
Name: 2, dtype: float64
```



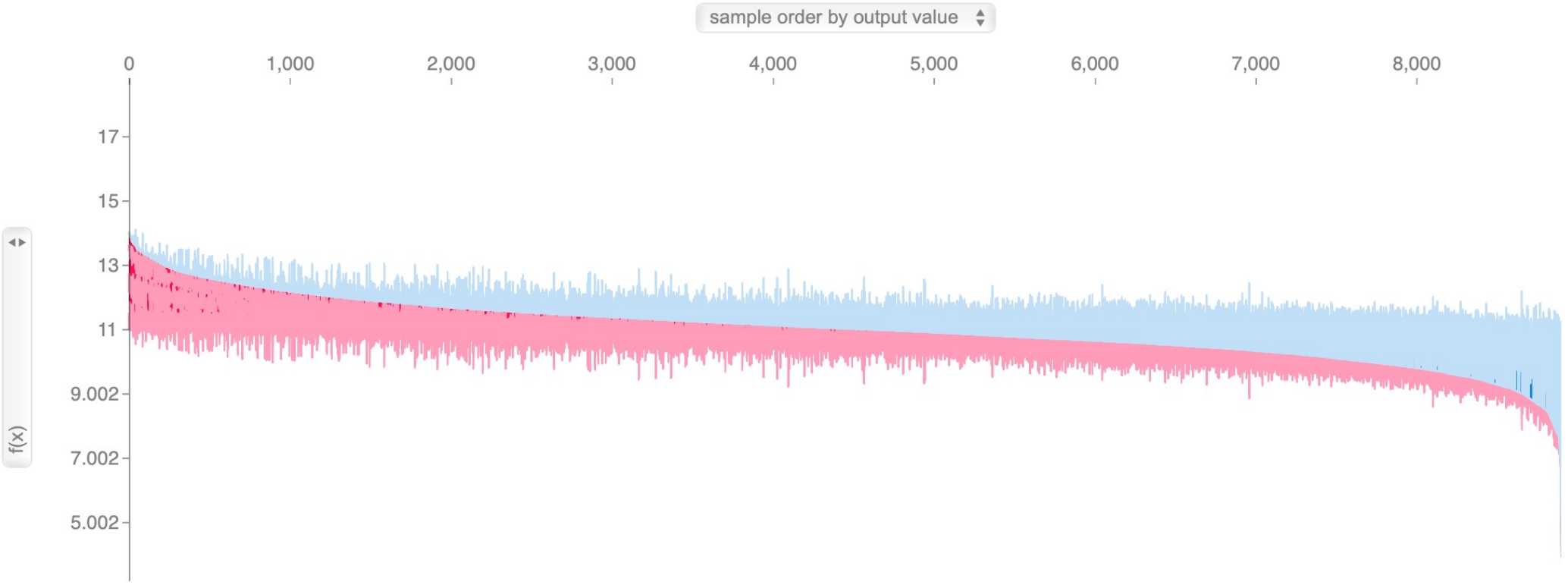
Global force plot

Sample ordered by similarity



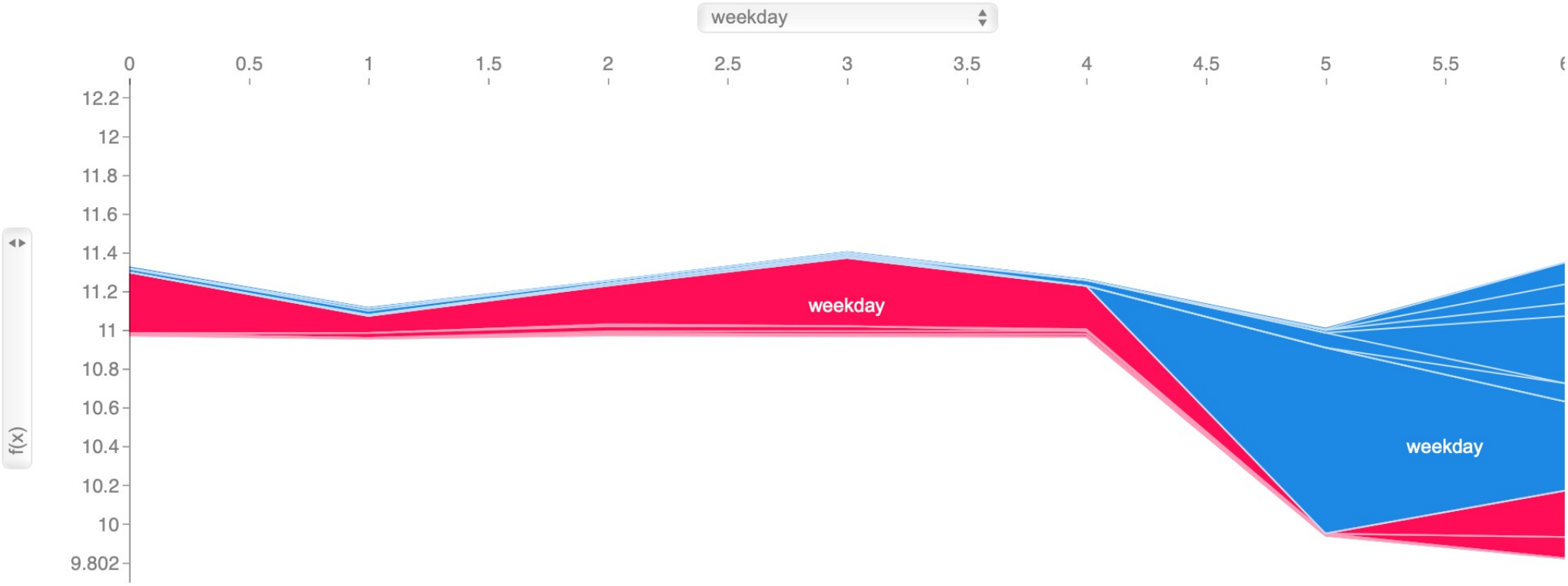
Global force plot

Sample ordered by output value



Global force plot

Weekday



Conclusion

Appendix

Code

```python
# Projet_final
import pandas as pd
import numpy as np
from sklearn.preprocessing import FunctionTransformer
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import KBinsDiscretizer
from sklearn.pipeline import make_pipeline
from xgboost import XGBRegressor
from sklearn.preprocessing import OneHotEncoder

def _encode_dates(X):
    # Make sure that DateOfDeparture is of dtype datetime
    X = X.copy()  # modify a copy of X
    X.loc[:, "DateOfDeparture"] = pd.to_datetime(X['DateOfDeparture'])
    # Encode the date information from the DateOfDeparture columns
    X.loc[:, 'month'] = X['DateOfDeparture'].dt.month
    X.loc[:, 'day'] = X['DateOfDeparture'].dt.day
    X.loc[:, 'weekday'] = X['DateOfDeparture'].dt.weekday
    X.loc[:, 'week'] = X['DateOfDeparture'].dt.week
    X.loc[:, 'n_days'] = X['DateOfDeparture'].apply(
        lambda date: (date - pd.to_datetime("1970-01-01")).days
    )

    #ADD Holidays
    we = [4, 5, 6]
    ho_we = [51, 35, 21, 26, 46]
    ho = [1, 52, 27, 47]
    ho_0 = [36, 22]
    summer = range(22, 35)
    X.loc[:, 'holiday'] = np.where((X['week'].isin(ho)) |
                                   ((X['week'].isin(ho_we)) & (X['weekday'].isin(we))) |
                                   ((X['week'].isin(ho_0)) & (X['weekday'] == 0)) |
                                  ((X['week'].isin(summer)) & (X['weekday'] == 5))
                                  , 1, 0)
    # Finally we can drop the original columns from the dataframe
    return X.drop(columns=["DateOfDeparture", "std_wtd"])
```

```python
def get_estimator():
    date_encoder = FunctionTransformer(_encode_dates)

    categorical_encoder = OneHotEncoder(sparse=False)
    categorical_cols = ["Arrival", "Departure"]

    numerical_cols = ['WeeksToDeparture']
    numerical_encoder = KBinsDiscretizer(n_bins=9,
                                         encode='ordinal',
                                        strategy='kmeans')

    preprocessor1 = make_column_transformer(
        (categorical_encoder, categorical_cols),
        (numerical_encoder, numerical_cols),
        remainder='passthrough'
        )


    regressor = XGBRegressor(colsample_bytree= 0.7,
                 learning_rate = 0.03,
                 max_depth = 10,
                 min_child_weight = 3, n_estimators = 2000,
                 early_stopping_rounds = 10,
                 nthread = 4, objective = 'reg:squarederror', subsample= 0.7)

    return make_pipeline(date_encoder, preprocessor1, regressor)
```