Predicting Bike Sharing Rental Demand

A group project by *Saurabh Chakravorty, Dipanshu Gupta, Johannes Hufeld, Benedikt Kirsch and Robert Maerz*

Intro to Data Analytics in Business - Prof. Peter Roßbach - MADS 2019

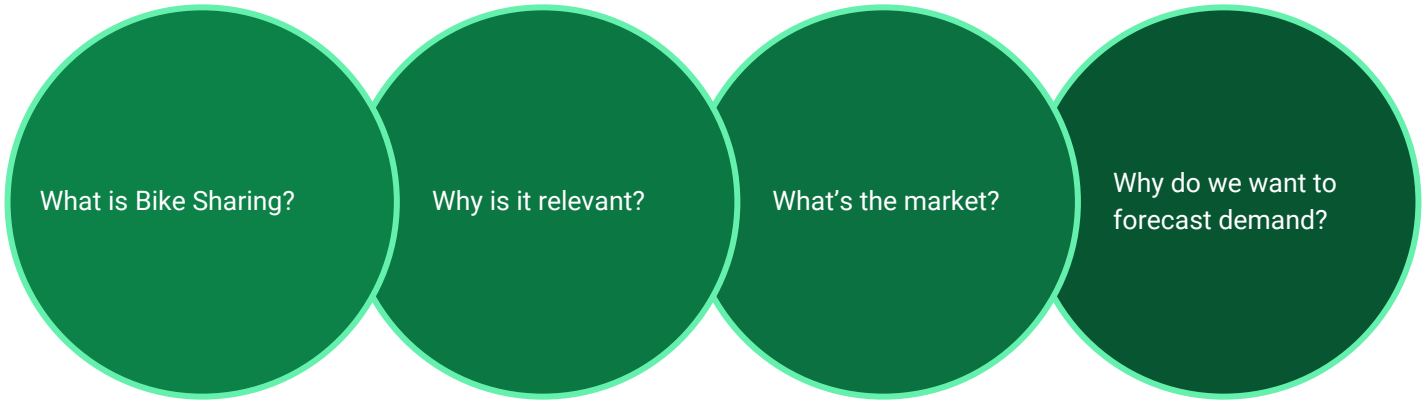
Agenda:

- Introduction
- Data Exploration
- Data Preparation
- Predictions & Results
- Conclusion

1. Bike Sharing Introduction

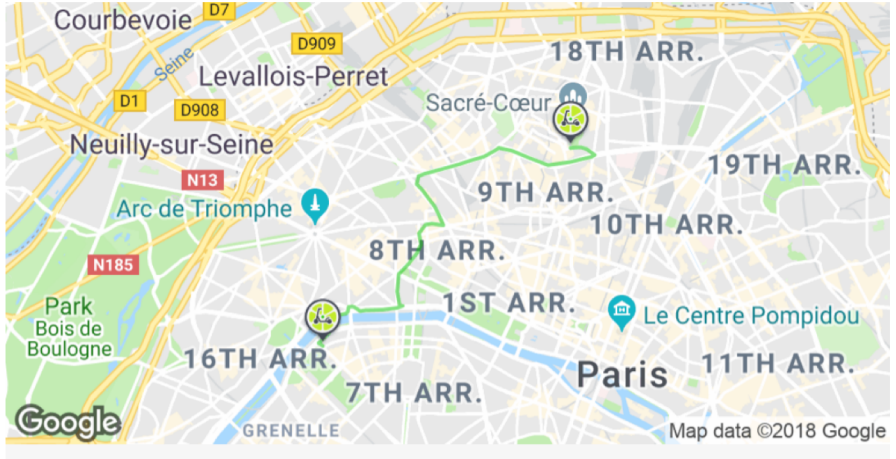
presented by Dipanshu

1. Bike Sharing Introduction



What is Bike Sharing?

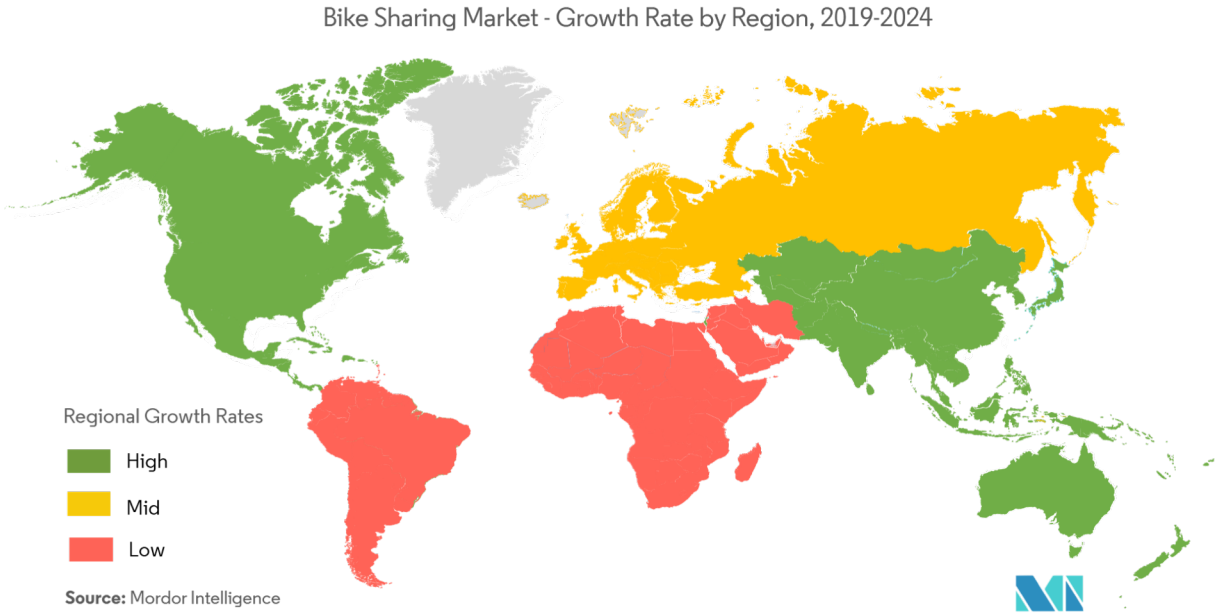
Why is it relevant?



6,8 Kilometer
Entfernung

0h48m
Zeit

What's the Market?



Why do we want to forecast Demand?

2. Data Exploration

presented by Robert and Benedikt

The Dataset

`df.head()`

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	16
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	13
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	1

- All columns except dteday appear to contain numerical values
- Values related to weather data (temp, atemp, hum, windspeed) might be scaled

The Dataset

- We are indeed dealing with numerical columns only (except dteday)
- Our dataset is 17379 rows long and 17 columns wide

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 17 columns):
instant       17379 non-null int64
dteday        17379 non-null object
season        17379 non-null int64
yr            17379 non-null int64
mnth          17379 non-null int64
hr            17379 non-null int64
holiday       17379 non-null int64
weekday       17379 non-null int64
workingday    17379 non-null int64
weathersit    17379 non-null int64
temp          17379 non-null float64
atemp         17379 non-null float64
hum           17379 non-null float64
windspeed     17379 non-null float64
casual        17379 non-null int64
registered    17379 non-null int64
cnt           17379 non-null int64
dtypes: float64(4), int64(12), object(1)
memory usage: 2.3+ MB
```

Missing Values

- We have no NaN values!
- We won't have to clean the data, will we?

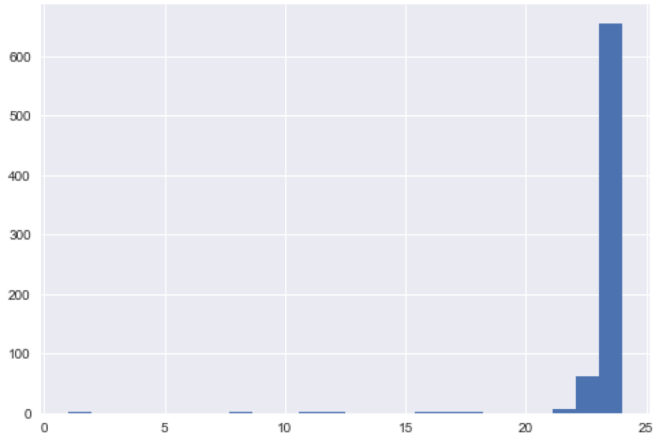
```
df.isna().sum()
```

```
instant       0
dteday        0
season        0
yr            0
mnth          0
hr            0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

Missing Values

```
hours_per_day = df.groupby('dteday').count()['instant'].hist(bins=24)
```

- Do some dates not have data for all 24 hours?



Missing Values

- We have no NaN values (check)
- We also have no zero values (check)
- Some days have missing hours (check)

- Dataset must be clean, as this means hours absent have zero rentals

```
0 in df.cnt.values
```

```
False
```

Statistics

`df.describe()`

	instant	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
count	17379.0000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000
mean	8690.0000	2.501640	0.502561	6.537775	11.546752	0.028770	3.003683	0.682721	1.425283	0.496987	0.475775	0.627229	0.190098	35.676218	153.786869	189.463088
std	5017.0295	1.106918	0.500008	3.438776	6.914405	0.167165	2.005771	0.465431	0.639357	0.192556	0.171850	0.192930	0.122340	49.305030	151.357286	181.387599
min	1.0000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.020000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	4345.5000	2.000000	0.000000	4.000000	6.000000	0.000000	1.000000	0.000000	1.000000	0.340000	0.333300	0.480000	0.104500	4.000000	34.000000	40.000000
50%	8690.0000	3.000000	1.000000	7.000000	12.000000	0.000000	3.000000	1.000000	1.000000	0.500000	0.484800	0.630000	0.194000	17.000000	115.000000	142.000000
75%	13034.5000	3.000000	1.000000	10.000000	18.000000	0.000000	5.000000	1.000000	2.000000	0.660000	0.621200	0.780000	0.253700	48.000000	220.000000	281.000000
max	17379.0000	4.000000	1.000000	12.000000	23.000000	1.000000	6.000000	1.000000	4.000000	1.000000	1.000000	1.000000	0.850700	367.000000	886.000000	977.000000

- Data looks to be real (2 years, 4 seasons, 12 months, 7 weekdays, 24 hours)
- We see some booleans (holiday, workingday); temp does not range from 0 to 1

Statistics

```
for (columnName, columnData) in df.iteritems():
    print(f'Column Name: {columnName}')
    print(columnData.value_counts().nlargest(24).sort_index())
```

- Count values for each index; print for all columns and sort by index
- Get an overview of what data looks like in each column

Statistics

- Warmer seasons (2 - Spring, 3 - Summer) have more data
- Years are quite close

```
Column Name: season
1    4242
2    4409
3    4496
4    4232
Name: season, dtype: int64


Column Name: yr
1    8734
0    8645
Name: yr, dtype: int64
```

Statistics

- Months are also close, with the exception of February (28 days in the month?)

```
Column Name: mnth
1     1429
2     1341
3     1473
4     1437
5     1488
6     1440
7     1488
8     1475
9     1437
10    1451
11    1437
12    1483
Name: mnth, dtype: int64
```

Statistics

- Days of the week are comfortably close to one another
- Weathersit sees a heavy imbalance we'll have to account for!

```
Column Name: weekday
0    2502
1    2479
2    2453
3    2475
4    2471
5    2487
6    2512
Name: weekday, dtype: int64


Column Name: weathersit
1    11413
2     4544
3     1419
4        3
Name: weathersit, dtype: int64
```

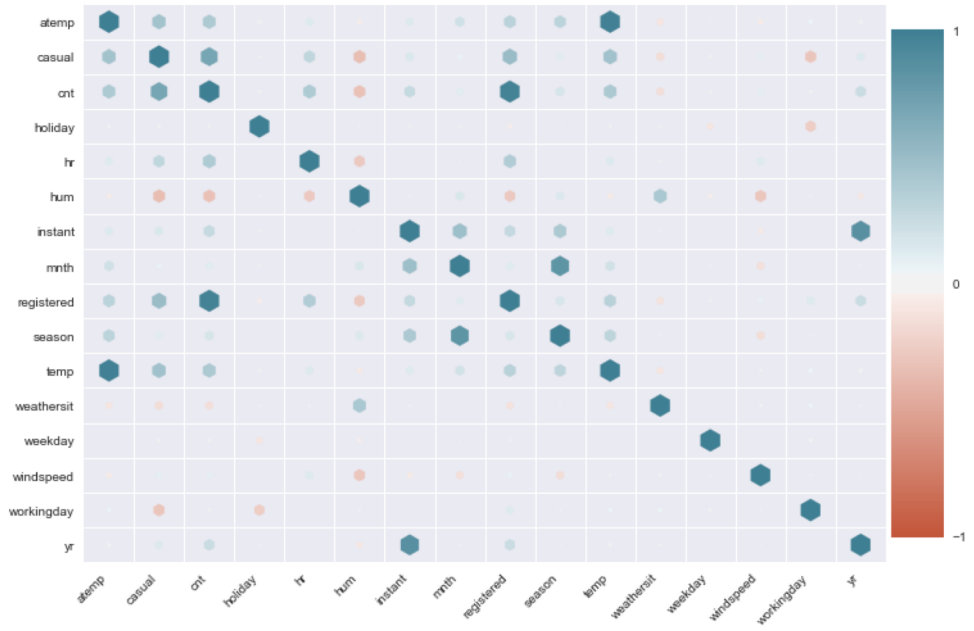
Statistics

- Hour counts peak at 4 PM with data being fairly uniform between 7 AM and 11 PM
- Data shows that people ride less bikes at night

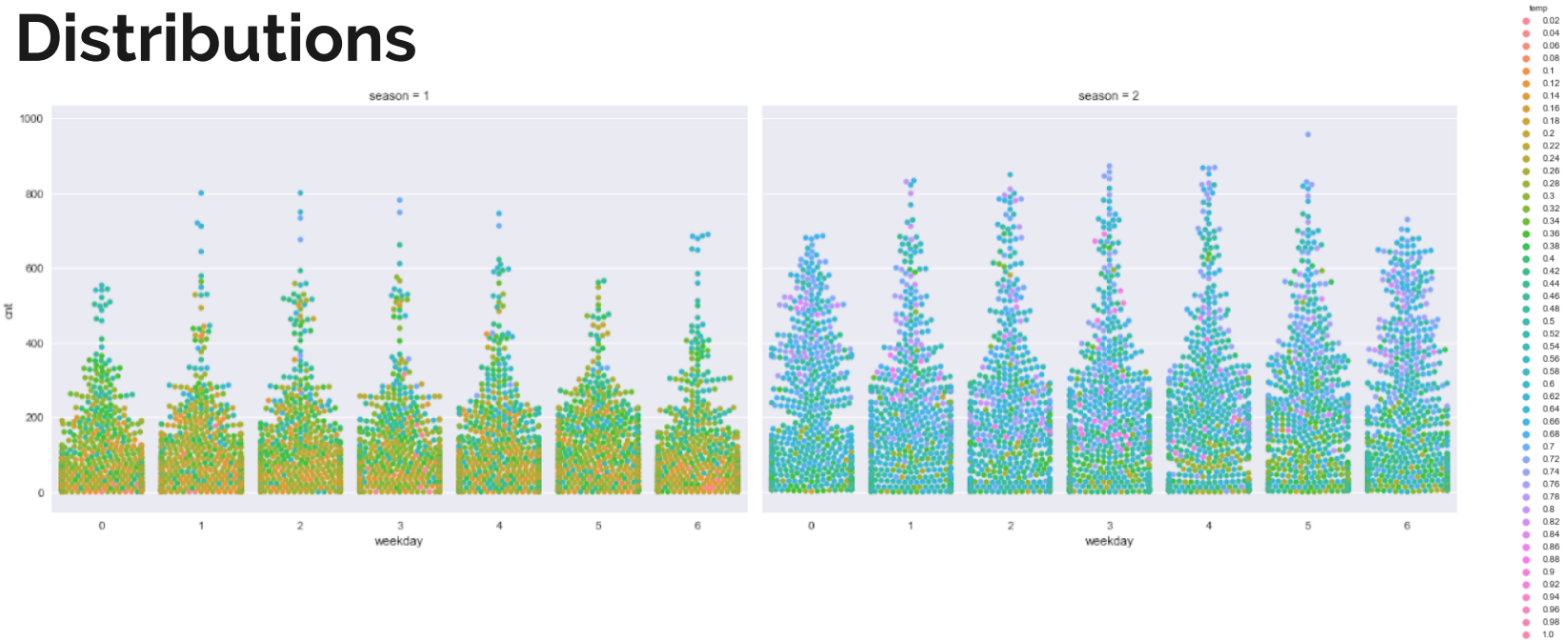
```
Column Name: hr
0     726
1     724
2     715
3     697
4     697
5     717
6     725
7     727
8     727
9     727
10    727
11    727
12    728
13    729
14    729
15    729
16    730
17    730
18    728
19    728
20    728
21    728
22    728
23    728
```

Correlation

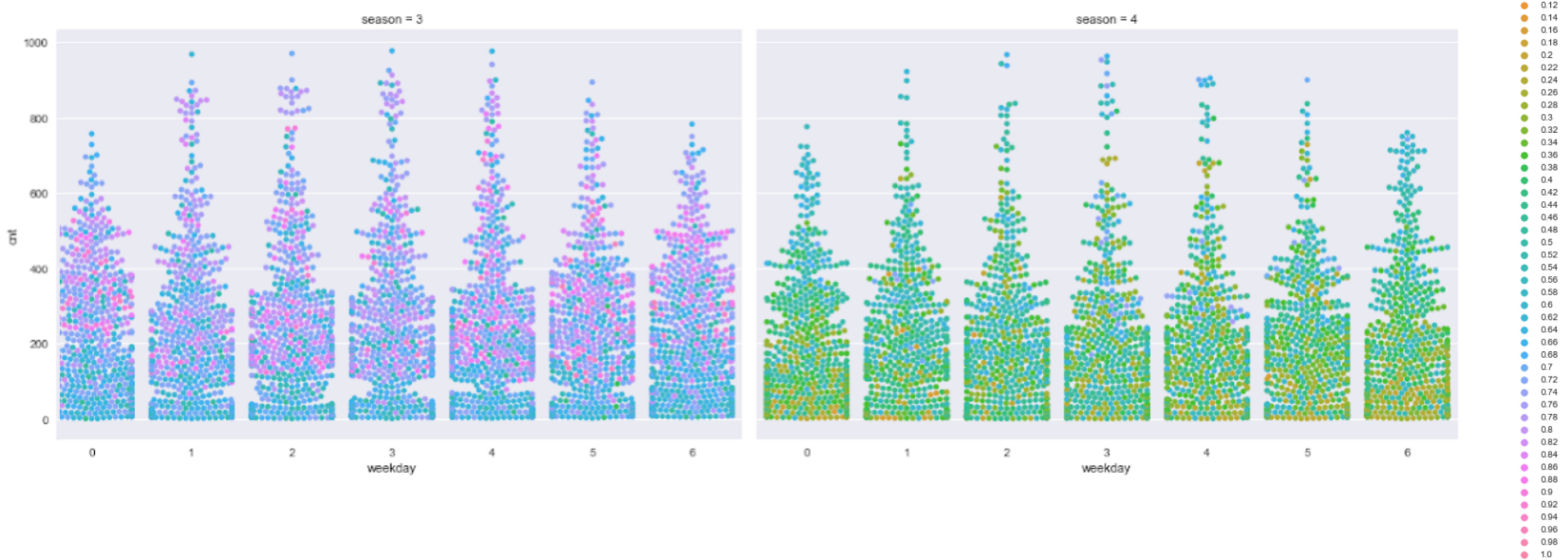
- Icon size and opacity indicate how strong the correlation is
- We see a lot more (strong) positive correlation than negatives
- Surprise: registered users have a high impact on total count
- Not a surprise: temp and atemp are correlated, just like season and month



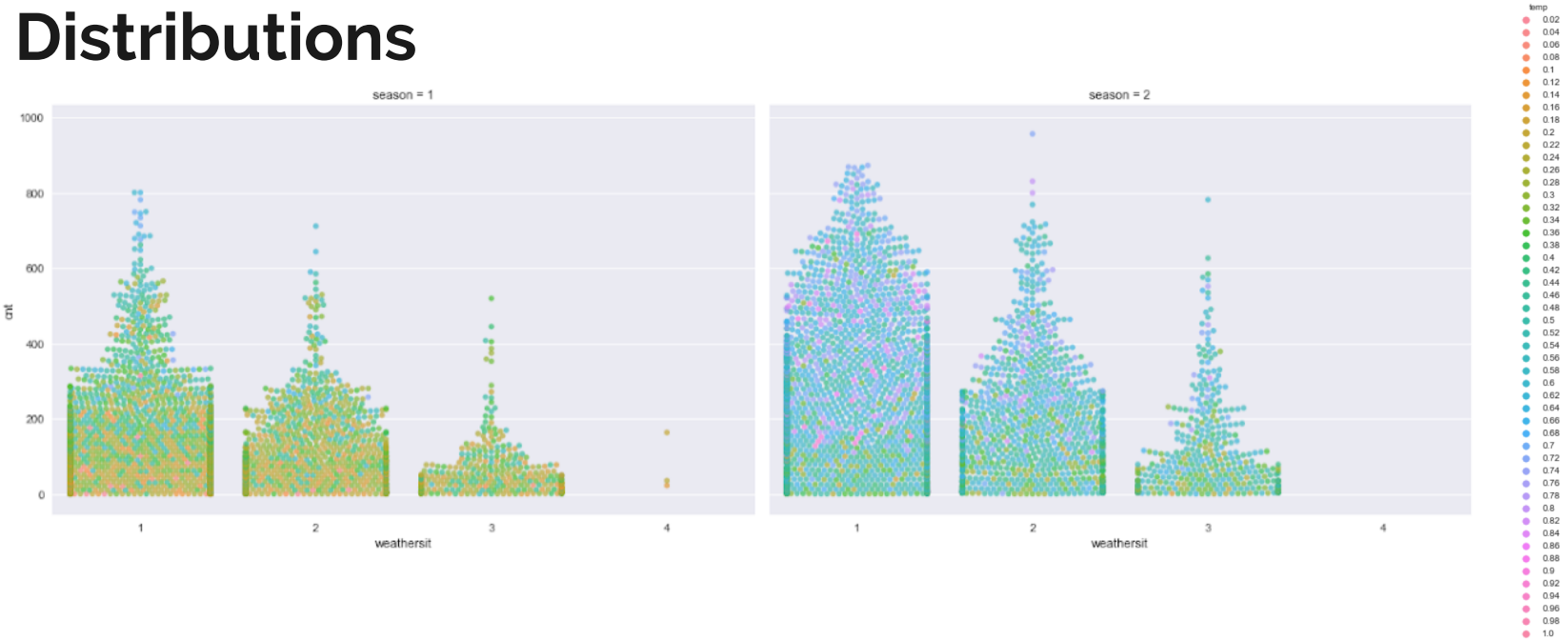
Distributions



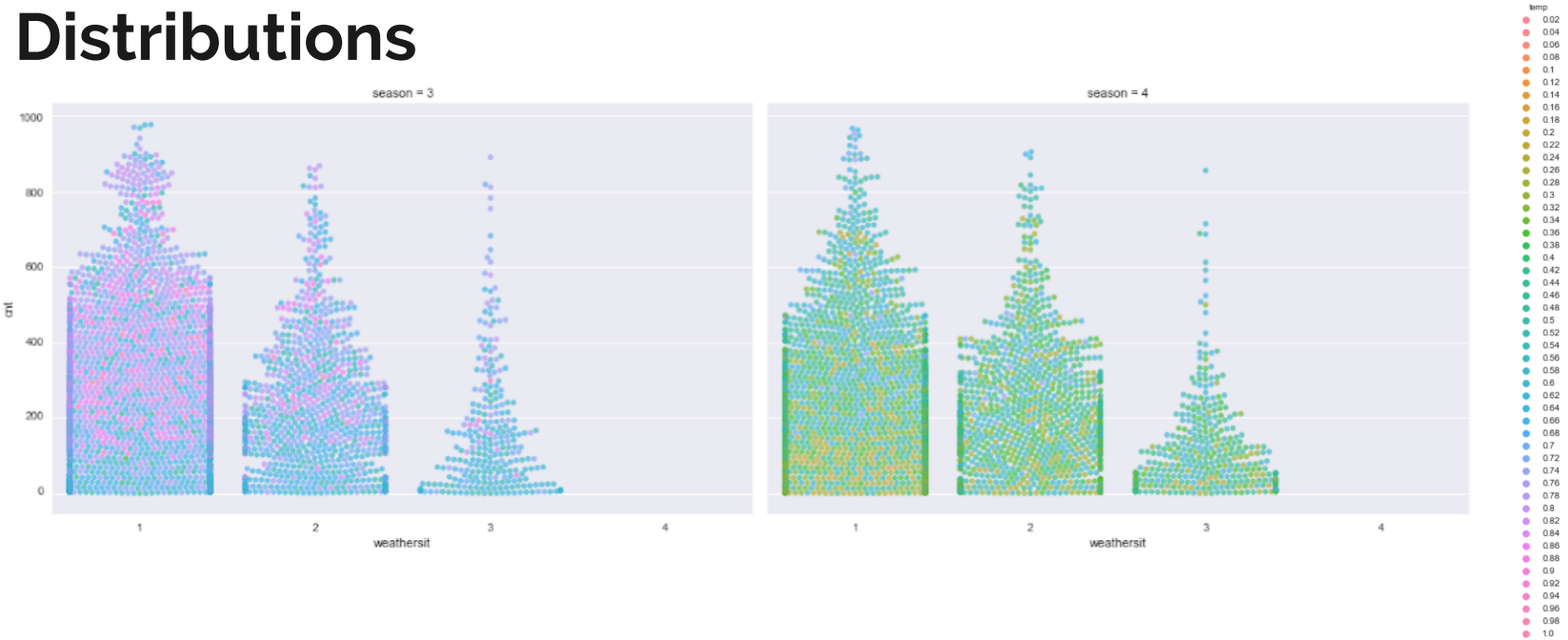
Distributions



Distributions



Distributions



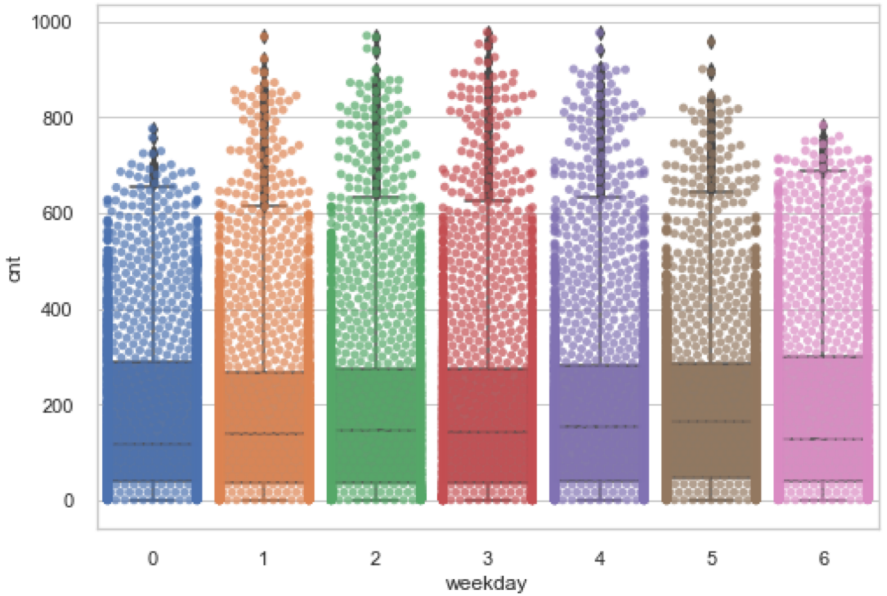
Outliers

- Outliers are data points that differ greatly from the rest of the data set, fall outside of:

$$[Q_{0.25} - k * (Q_{0.75} - Q_{0.25}), Q_{0.75} + k * (Q_{0.75} - Q_{0.25})], k \in [1.5, 3]$$

- Remove extreme outliers?
- Determine % share of extreme observations in data:

User type	k = 1.5	k = 3
Casual	6.9%	2.8%
Registered	3.2%	0.3%
Total	2.9%	0.0%

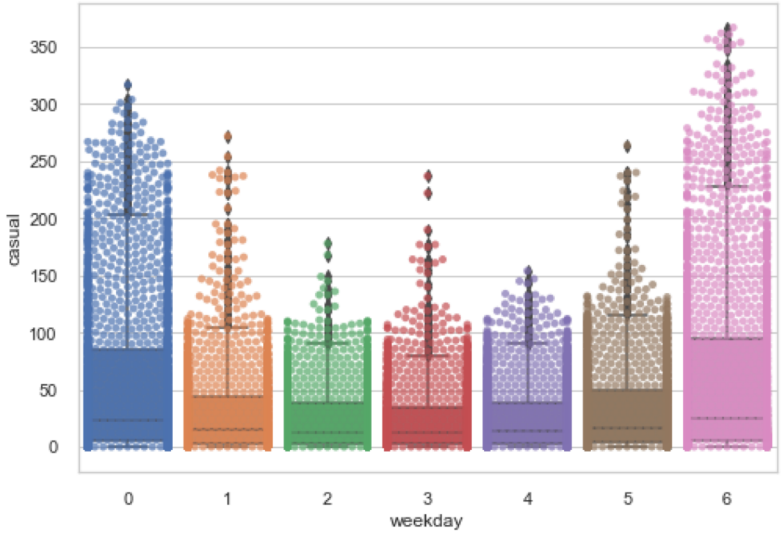


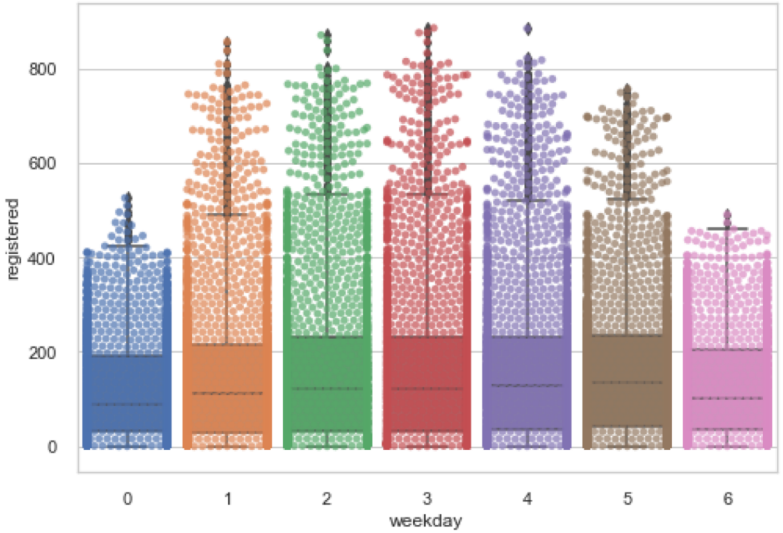
Outliers

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
15588	15589	10/16/2012	4	1	10	17	0	2	1	1	0.52	0.5	0.39	0.194	104	839	943
15444	15445	10/10/2012	4	1	10	17	0	3	1	1	0.58	0.5455	0.43	0.2239	91	857	948
15108	15109	9/26/2012	4	1	9	17	0	3	1	1	0.74	0.6667	0.48	0.2985	77	876	953
10622	10623	3/23/2012	2	1	3	17	0	5	1	2	0.72	0.6515	0.42	0.1642	264	693	957
15780	15781	10/24/2012	4	1	10	17	0	3	1	1	0.66	0.6212	0.47	0	87	876	963
15084	15085	9/25/2012	4	1	9	17	0	2	1	1	0.66	0.6212	0.39	0.2836	107	860	967
14725	14726	9/10/2012	3	1	9	18	0	1	1	1	0.62	0.6212	0.35	0.2985	111	857	968
14748	14749	9/11/2012	3	1	9	17	0	2	1	1	0.7	0.6364	0.28	0	168	802	970
14964	14965	9/20/2012	3	1	9	17	0	4	1	1	0.64	0.6212	0.5	0.2239	91	885	976
14773	14774	9/12/2012	3	1	9	18	0	3	1	1	0.66	0.6212	0.44	0.2537	91	886	977

- Subjective evaluation of largest outliers (sanity checks) → keep them

Outliers

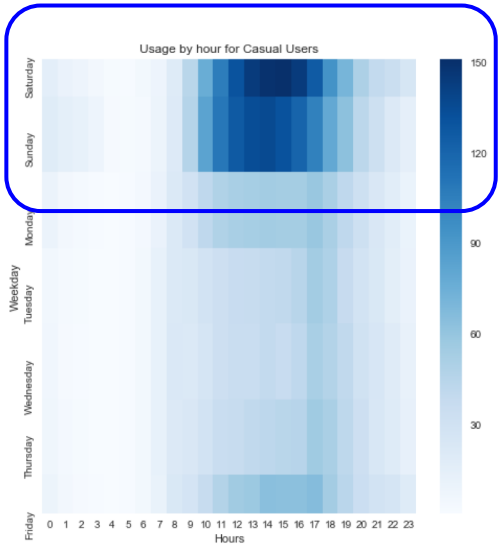


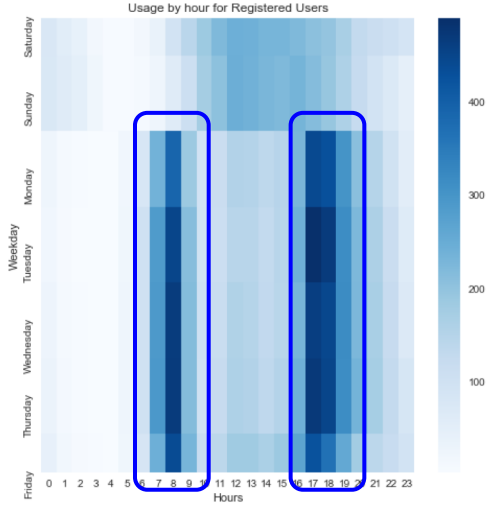


- Casual user (LHS) vs. registered user (RHS) data by weekday on box plot with swarmplot overlay
- Outlier behavior for both groups differs (more for casual), but overall demand patterns look different...

Heatmaps

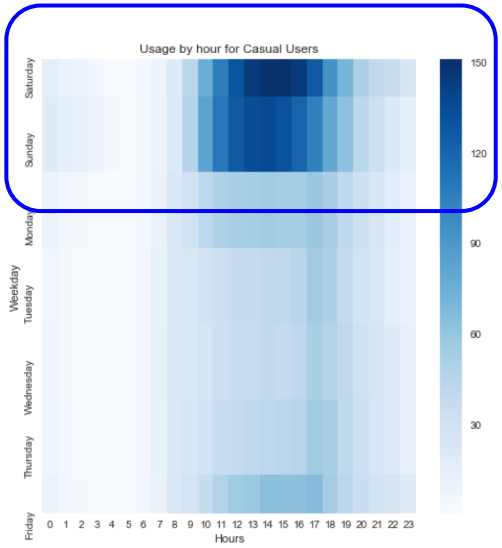
- Heatmaps depicting bike user by weekday by hour of the day
- Distinctly different behavior between registered and casual users
- Registered users: usage peaks during rush hour on weekdays

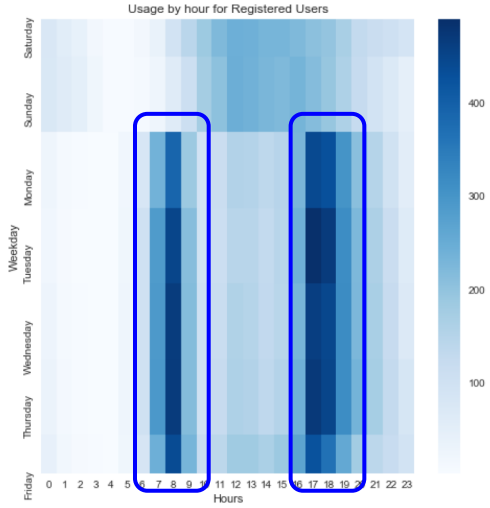




Heatmaps

- Casual users: peaks around midday during the weekend
- **Hypothesis**: split dataset in two and perform predictions on separate sets (combine later and compare)

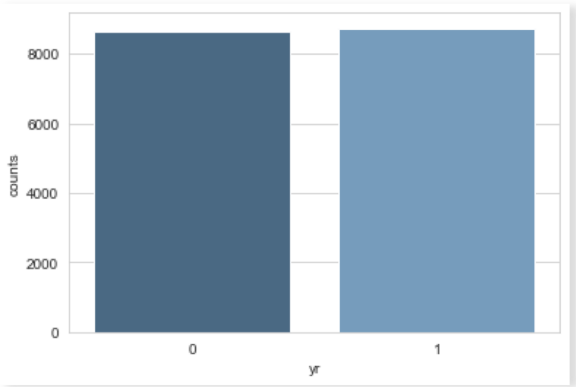




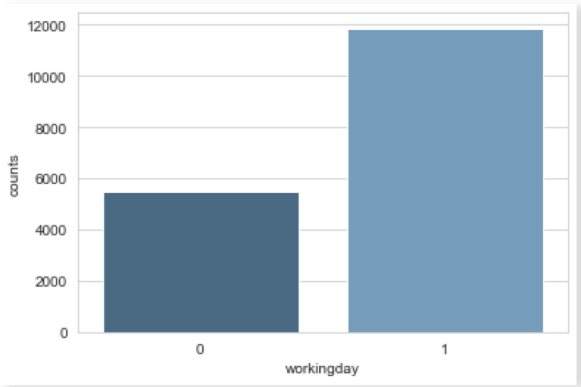
3. Data Preparation

presented by Johannes

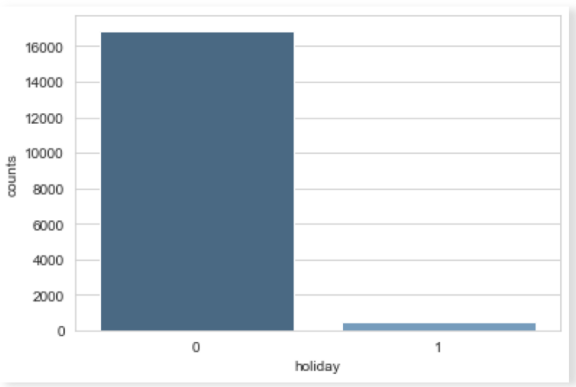
3.1.1 Unmodified Variables - Binary



year



working-day

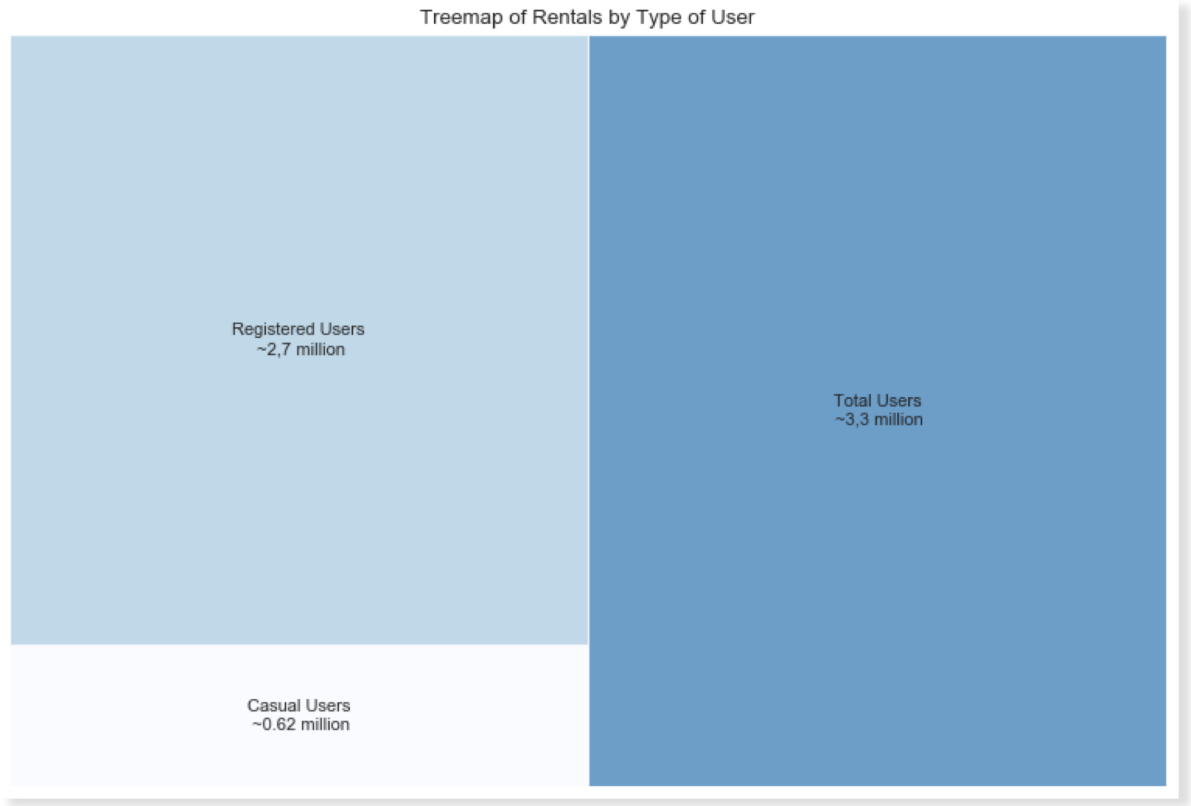


holiday

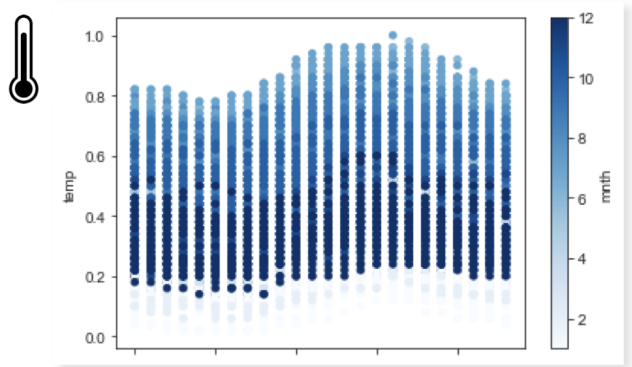
3.1.2 Unmodified Variables - Users

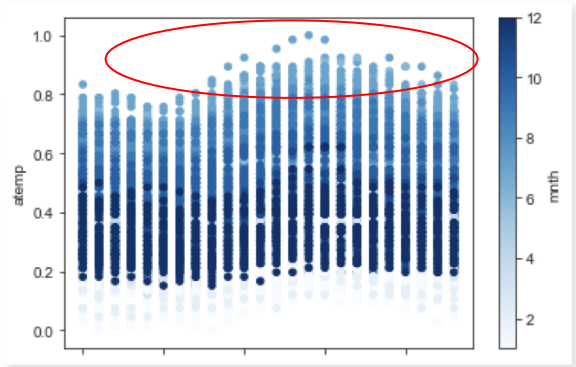
Users which we will attempt to **predict**.

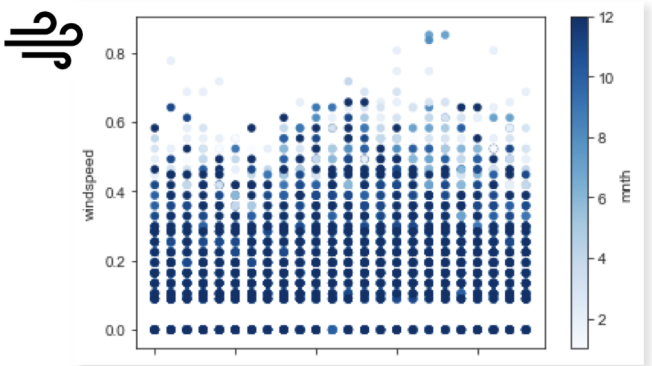
Total users is the sum of **registered** and **casual** users.

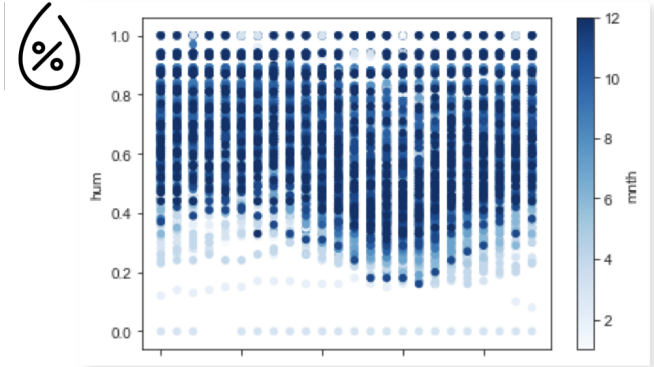


3.1.3 Pre-Normalized Weather Data

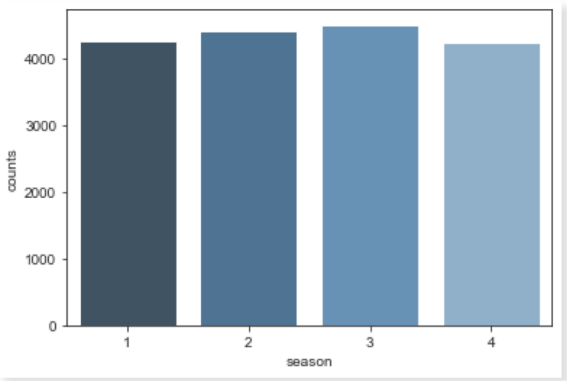


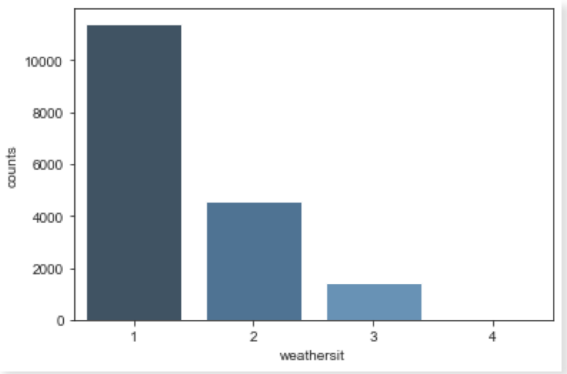


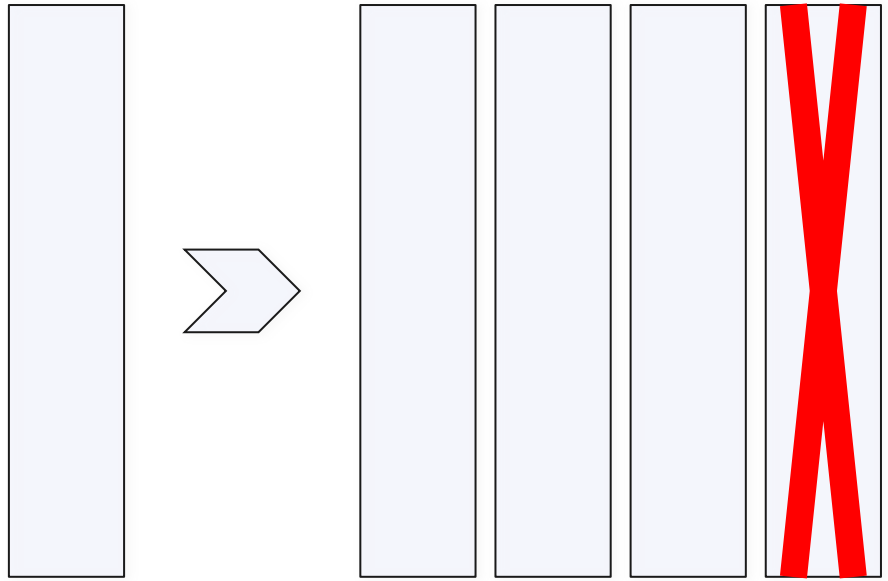




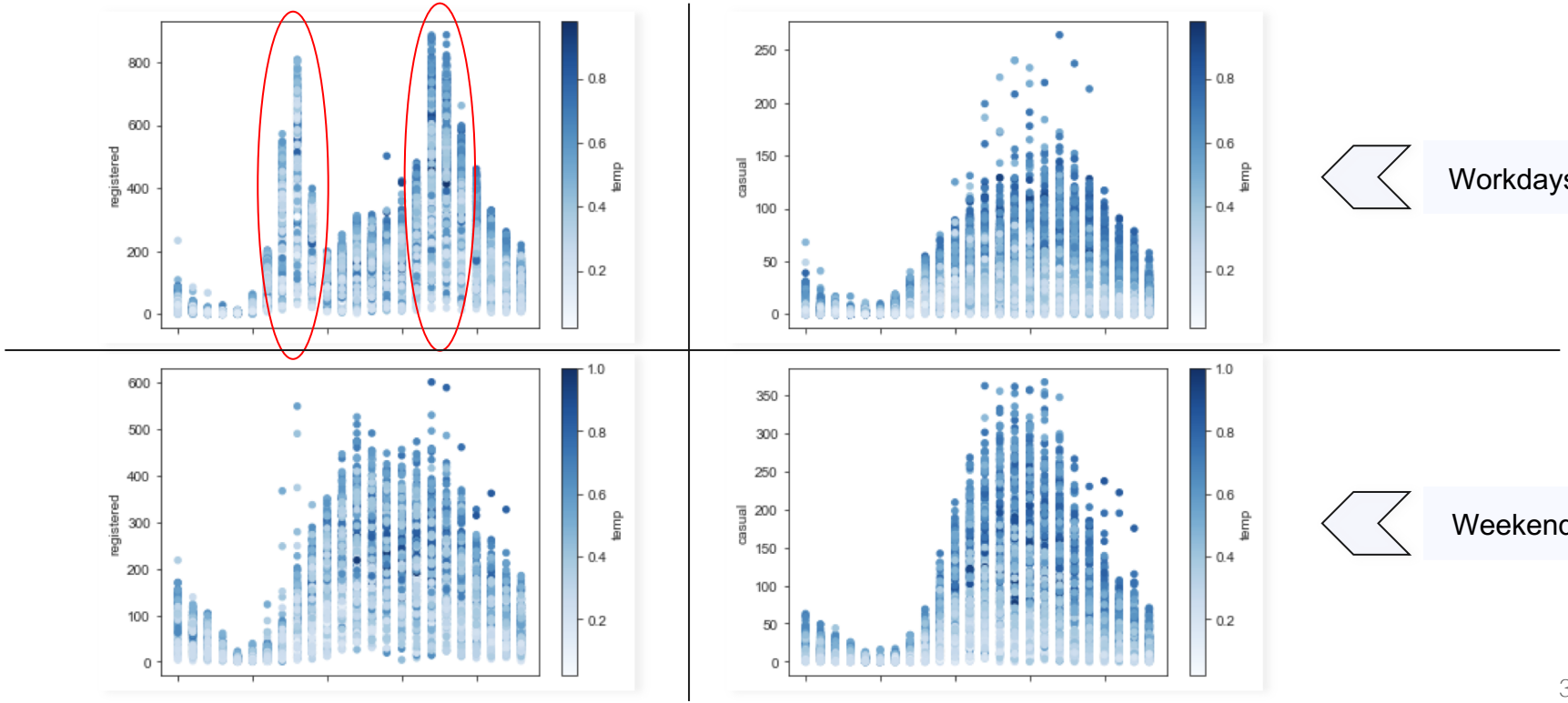
3.2 One-Hot Encoded Variables



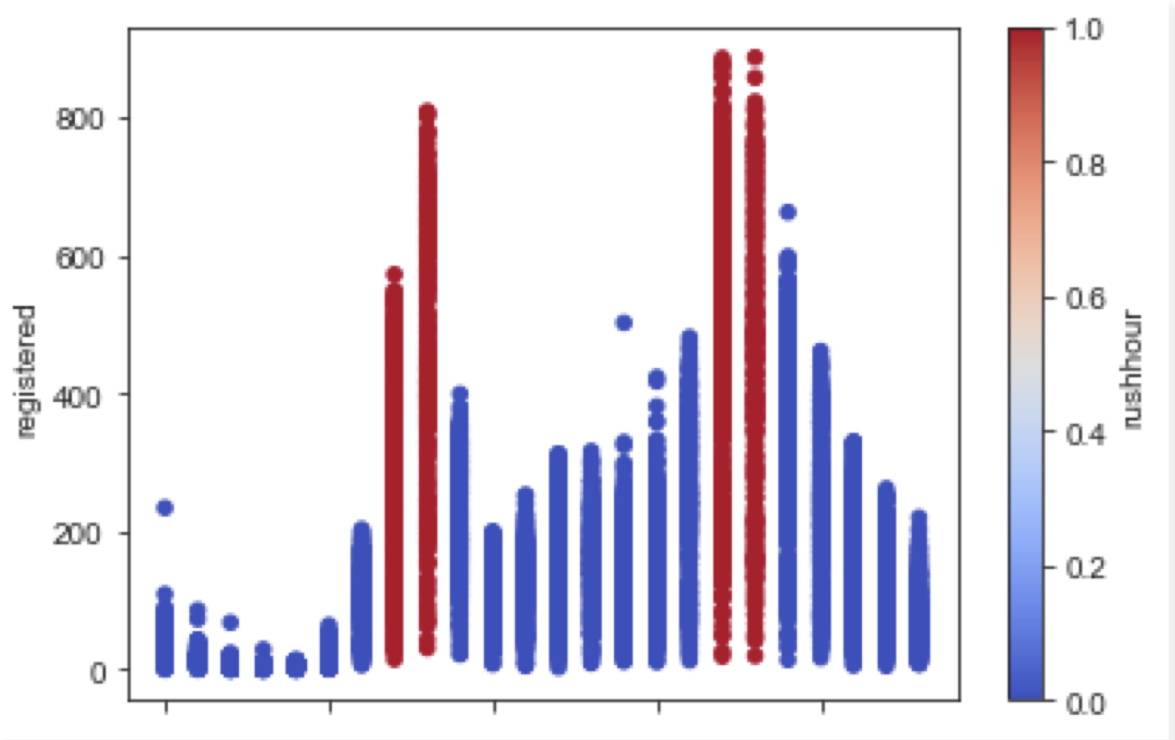




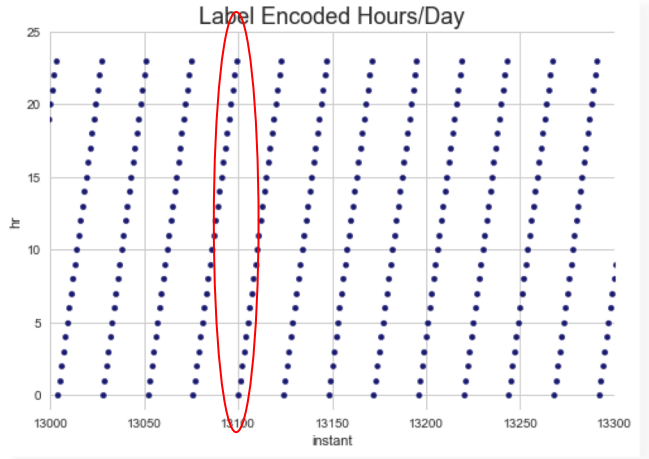
3.3 One of these is not like the others:

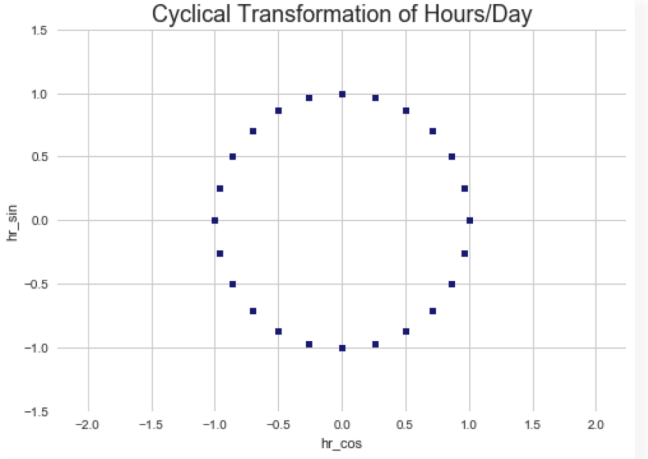


3.3.2 Rush Hour Feature:



3.4.1 Cyclical-Transformed Variable

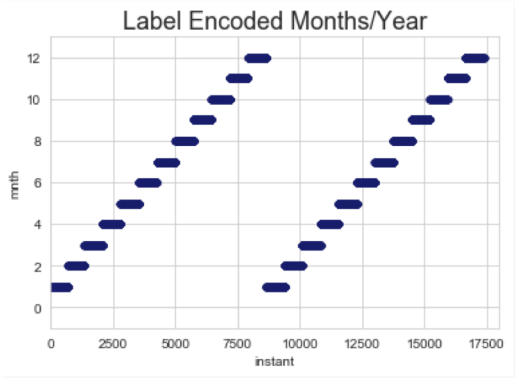


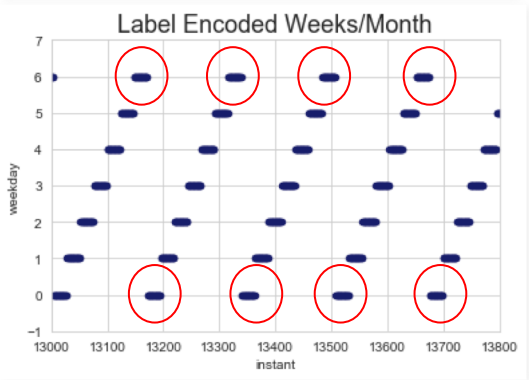


Large distance between time points

Solution: sine and cosine features

3.4.2 Months and Weeks?





We tried plenty: cyclical transformation, one-hot encoding, scaling, any combination of the previous techniques, it did not matter.

Doing nothing came out on top.

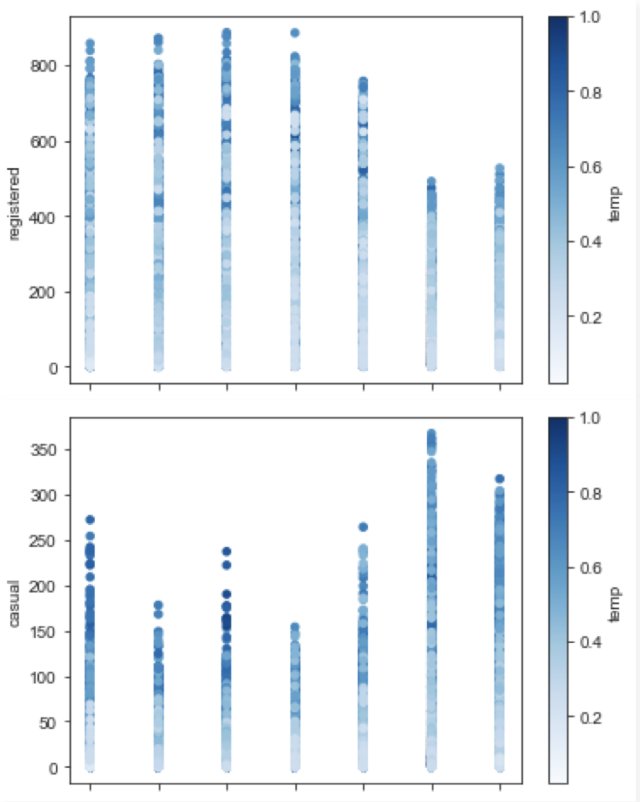
Except for one small change:

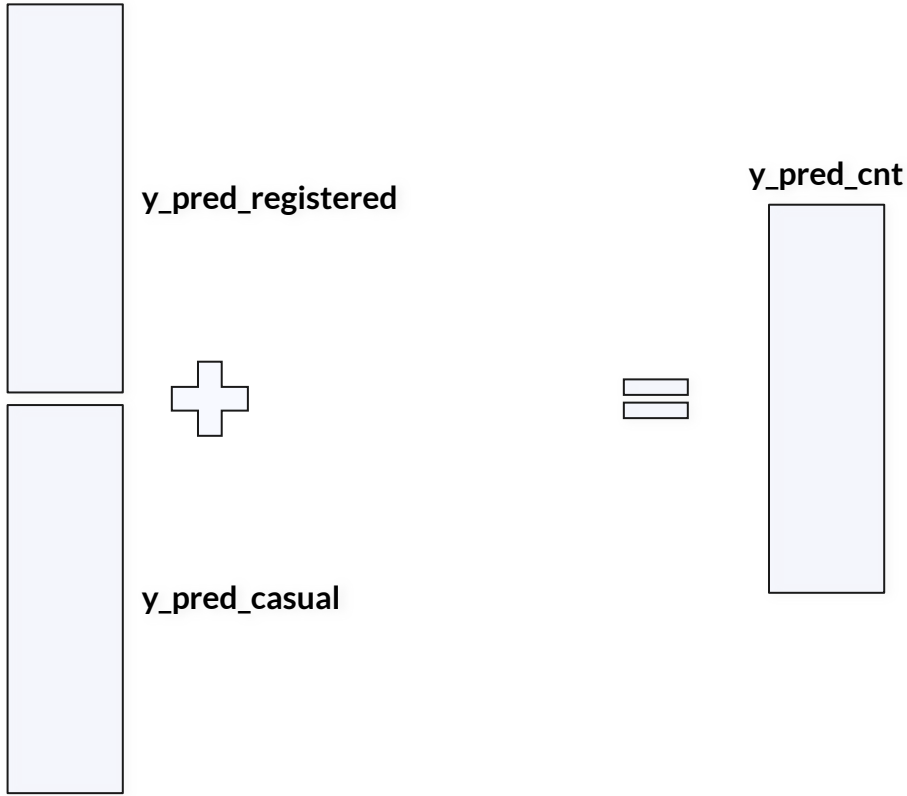
```
df['weekday'][df['weekday'] == 0] = 7
```

3.5 Dropped Variables

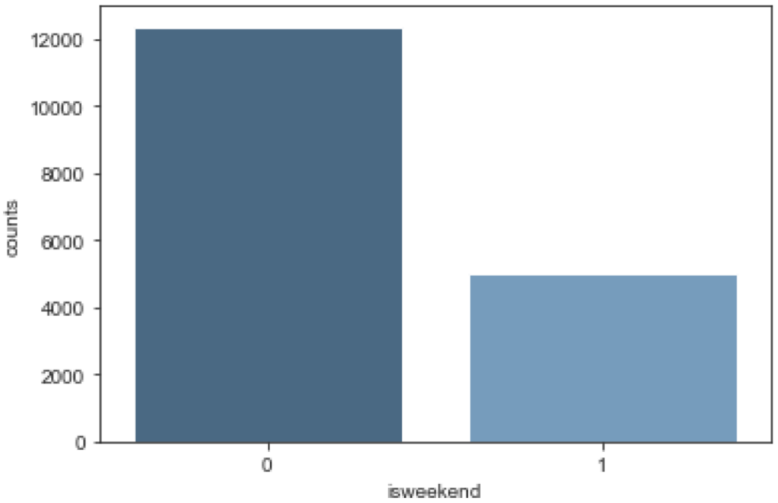
Unused Variables	Transformation Scrap	Target Variables
dteday	hr	cnt
instant	season_4	casual & registered
	weathersit_3	

3.6 Data Split by Users





3.6 Data Split by Weekday - same intuition



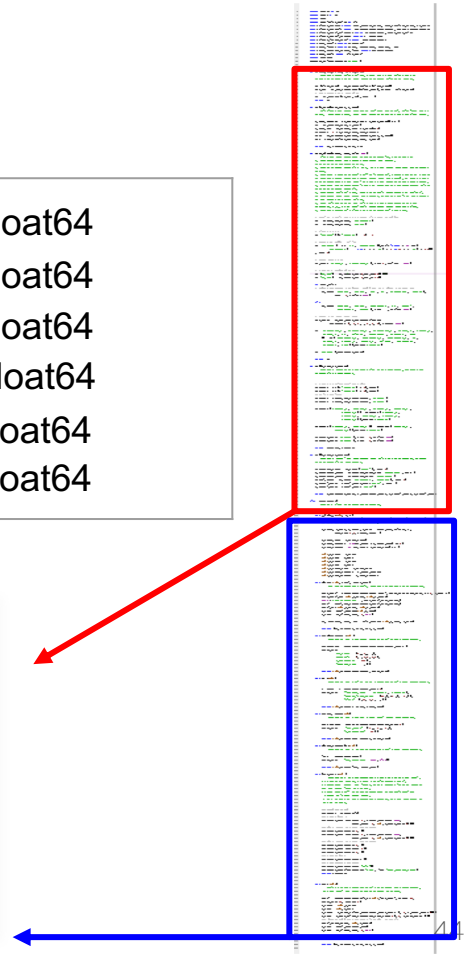
	Casual	Registered
	~12.000	~12.000
	~5.000	~5.000

3.8 Post-Prep. Variables

season_1	category	yr	category	temp	float64
season_2	category	mnth	category	atemp	float64
season_3	category	holiday	category	hum	float64
weathersit_2	category	weekday	category	windspeed	float64
weathersit_1	category	workingday	category	hr_sin	float64
weathersit_4	category	rushhour	category	hr_cos	float64

cnt
casual
registered

The entire data preparation and subsequent testing has become infinitely more **convenient** by **combining all elements** into one **importable library**. The **red** box covers all the code for **data preparation**, the **blue** box covers all **prediction models**.

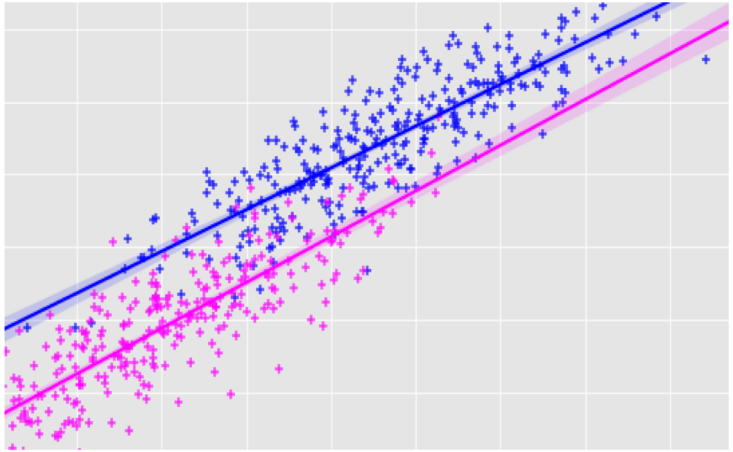


4. Forecasting

presented by Benedikt / Saurabh / Dipanshu

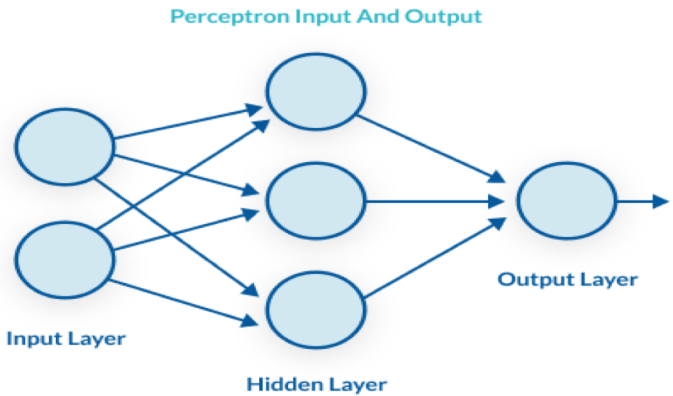
4.1 Linear Regression

- Simple approach to check the accuracy first in the linear world
- The model used here had absolutely no or miniscule correlation between independent variables
- The idea was to represent to get the best fit line among all the independent variables



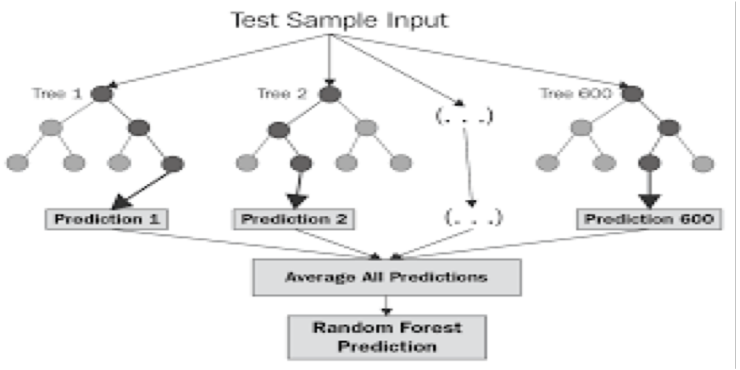
4.2 Multi-Layer Perceptron (MLP)

- Multi Layer Perceptron used here extends the concept of linear regression in neural network via using multiple perceptrons
- Training data is used for all the different scenarios and back propagation is done to adjust the weight and biases
- The main aim is to understand the causation of model output caused by this hidden layers via activation function



4.3 Random Forests

- The reason for selecting CART based algorithms is accuracy
- It uses classification trees with bagging technique where each tree is built independently
- The idea to intuitively use this algorithm is observation of less correlation between variables and having a clear deterministic factor
- The results of using this approach would be weighted sum of averages of all trees

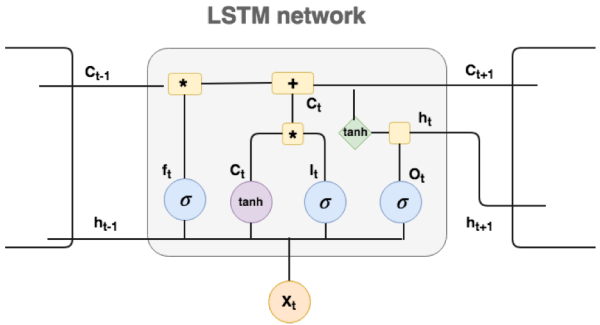


4.4 Gradient Boosting

- Gradient boost is based on the idea of weak learners which eliminates the concept of high bias as in RF
- The concept of regression trees is used here where each tree is built by boosting and eliminating errors by gradient descent
- Boosting technique used here yield the best results for all the scenarios of modelling
- R-Squared score is high than rest in almost all cases

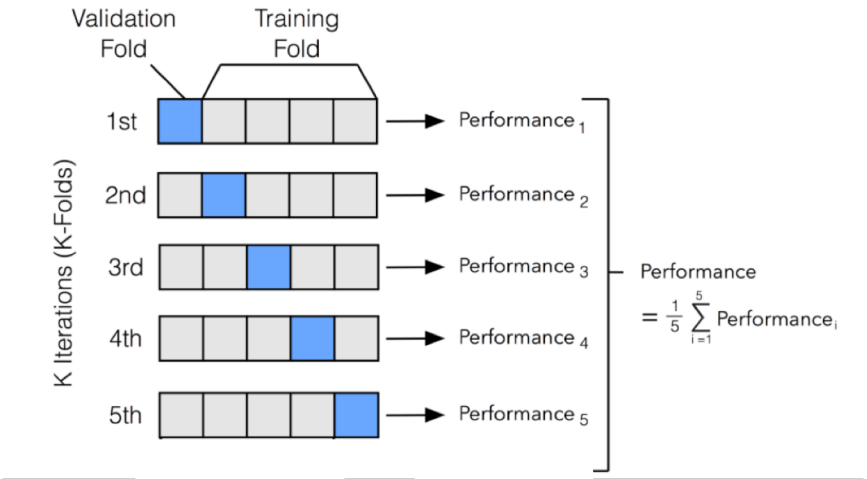
4.5 Long Short-Term Memory (LSTM)

- Simple neural networks cannot capture intertemporal dependencies, LSTM instead contains its information in a memory (gated cell: gates open/close to store information)
- Three-gate structure: input, forget and output gate
- Idea: sequential data where day and time of the day impact predictions, model performance probably superior to MLP



4.6 Cross Validation

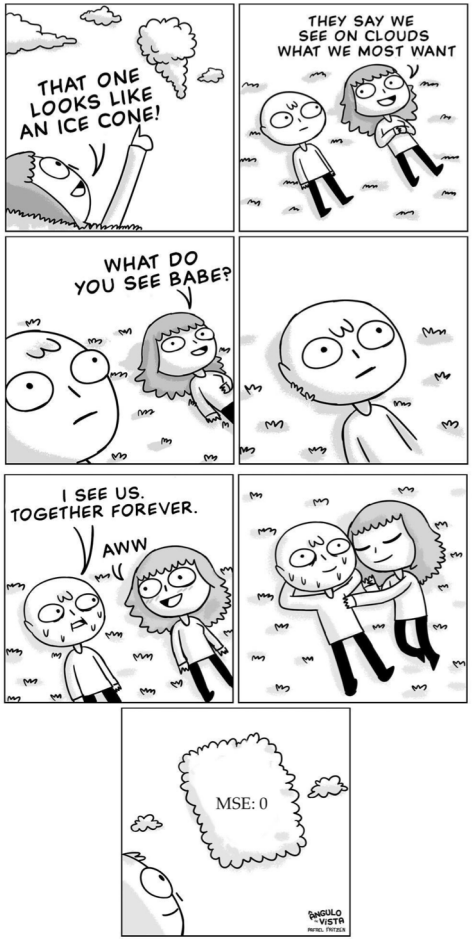
- Validation used here is K-Fold validation with k = 5 and train test split of 80 : 20
- **Grid Parameter Search** is used from Python where hyperparameter tuning was adjusted with various iterations



4.7 Evaluation Metrics

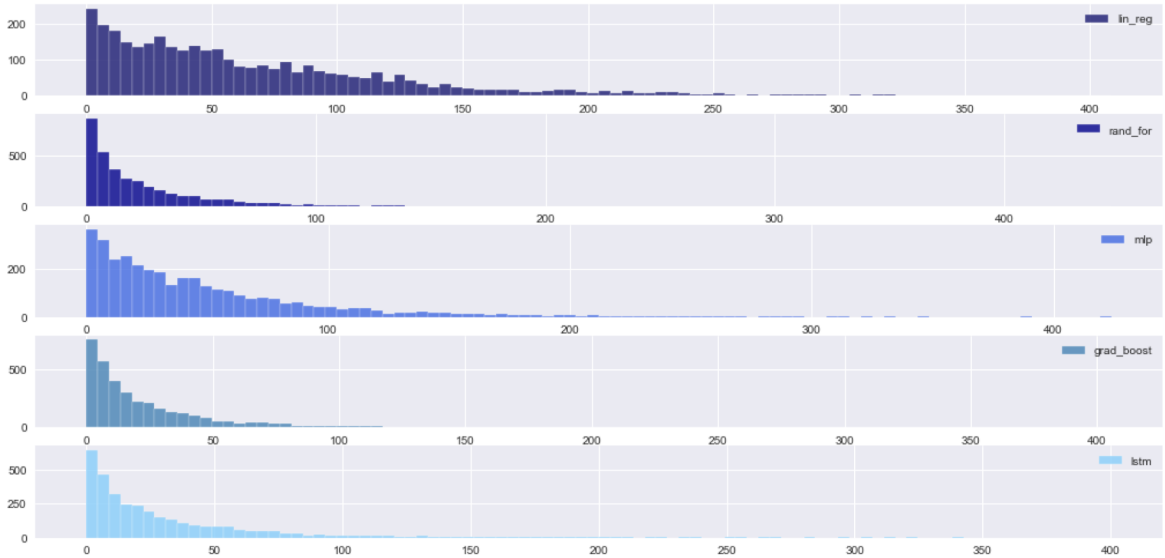
Parameters used here for evaluation in all models

1. R Squared
2. Pseudo R Squared
3. Mean Absolute Error(MAE)
4. Root Mean Square Absolute Error(RMSE)



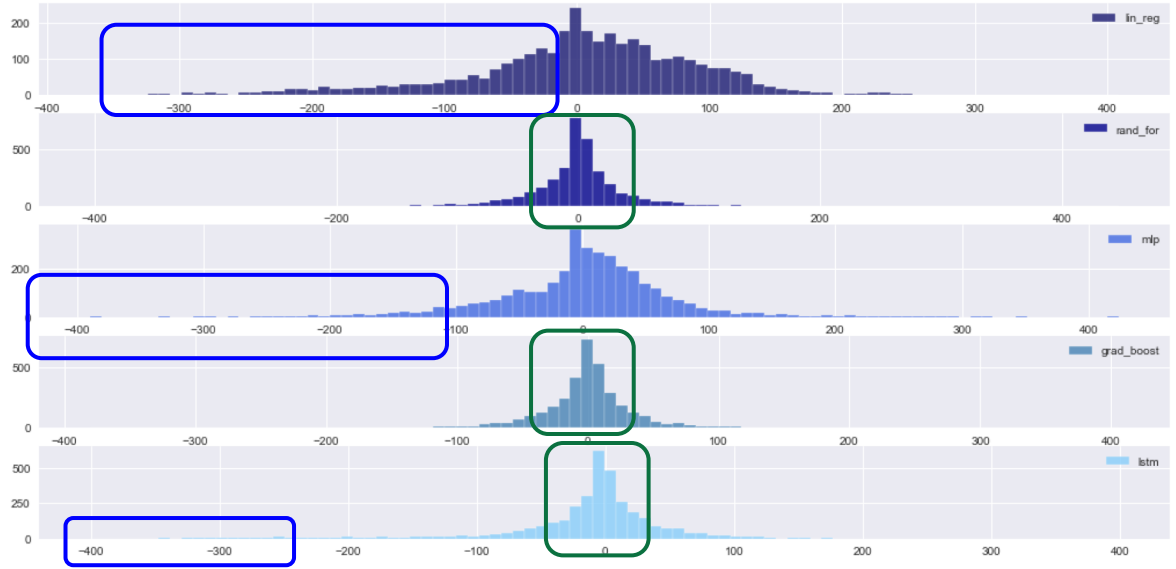
4.8 Error Evaluation

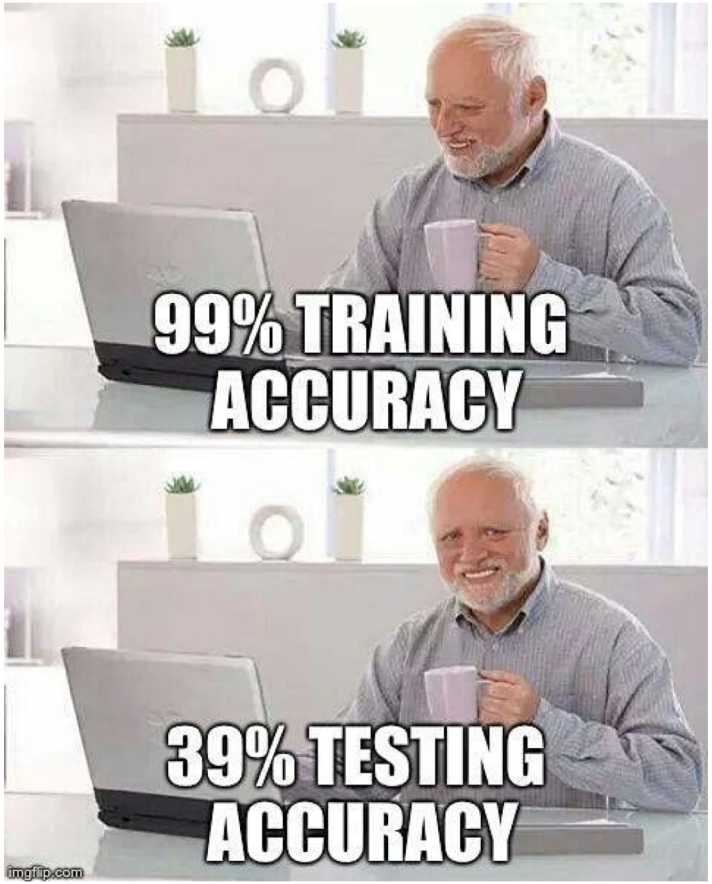
- MAE does not tell entire story
- Under- vs. overshooting



4.8 Error Evaluation

- Distribution of errors (y_pred vs. y_test)
- Lin. Reg worst outliers
- RF highest concentration around center (good)
- LSTM superior vs. MLP




99% TRAINING ACCURACY
39% TESTING ACCURACY
imgflip.com

4.9 Table of results

Model	User	R2 Score	Pseudo-R2 Score	RMSE	MAE
Linear Regression	Casual	73.8%	61.3%	77.86	55.54
	Registered	61.0%	7.4%	31.59	19.26
	Total	74.2%	60.0%	92.76	67.95
	No split	74.3%	60.7%	92.59	67.72
Random Forest	Casual	94.2%	93.6%	36.67	21.14
	Registered	90.6%	89.3%	15.52	8.90
	Total	94.5%	94.0%	42.82	25.36
	No split	94.2%	93.7%	43.82	26.12
MLP	Casual	85.9%	82.7%	57.14	39.93
	Registered	83.8%	79.6%	20.35	11.56
	Total	87.3%	84.6%	65.16	45.41
	No split	85.1%	81.5%	70.48	49.58
Gradient Boost	Casual	95.3%	95.0%	33.01	19.41
	Registered	92.0%	91.4%	14.32	8.43
	Total	95.4%	95.1%	39.22	23.48
	No split	95.4%	95.1%	39.30	23.88
LSTM	Casual	90.4%	87.2%	47.14	27.74
	Registered	87.9%	86.9%	17.63	9.74
	Total	91.4%	89.0%	53.71	32.36
	No split	90.2%	88.1%	57.20	34.18

4.9 Table of results

Model	User	R2 Score	Pseudo-R2 Score	RMSE	MAE
Linear Regression	Casual	73.8%	61.3%	77.86	55.54
	Registered	61.0%	7.4%	31.59	19.26
	Total	74.2%	60.0%	92.76	67.95
	No split	74.3%	60.7%	92.59	67.72
Random Forest	Casual	94.2%	93.6%	36.67	21.14
	Registered	90.6%	89.3%	15.52	8.90
	Total	94.5%	94.0%	42.82	25.36
	No split	94.2%	93.7%	43.82	26.12
MLP	Casual	85.9%	82.7%	57.14	39.93
	Registered	83.8%	79.6%	20.35	11.56
	Total	87.3%	84.6%	65.16	45.41
	No split	85.1%	81.5%	70.48	49.58
Gradient Boost	Casual	95.3%	95.0%	33.01	19.41
	Registered	92.0%	91.4%	14.32	8.43
	Total	95.4%	95.1%	39.22	23.48
	No split	95.4%	95.1%	39.30	23.88
LSTM	Casual	90.4%	87.2%	47.14	27.74
	Registered	87.9%	86.9%	17.63	9.74
	Total	91.4%	89.0%	53.71	32.36
	No split	90.2%	88.1%	57.20	34.18

5. Conclusion

presented by Johannes

5.1 Three key areas:

sporadic major errors

analyse error source

tune features and models

business implications

low & high predictions

rmsle

time management

model selection

moving forward

People with no idea about AI saying it will take over the world:

My Neural Network: