Bike Users

EDA and Predictions

Deloitte.

Presented by:
Nir Rozenstein

Date Presented:
January 5th, 2023

The business question:

1. Predict how many users will have for a given parameters

2. Which parameters have the most influence on the number of users

OUR TIMELINE

- EDA
- Insights
- Feature Engineering
- Build & Evaluate Models
- Conclution

Deloitte.

EDA

Exploratory Data Analysis

Deloitte.

Data Overview

Upload and review the data

In [103]:
```
main_original_df = pd.read_excel(r'C:\Users\nirro\Downloads\deloitte\Bike.xlsx',sheet_name='Data')
main_dict = pd.read_excel(r'C:\Users\nirro\Downloads\deloitte\Bike.xlsx',sheet_name='Dic')
```

In [104]:
```
main_original_df.head(100)
```

Out[104]:

	instant	dteday	season	yr	mnth	hr	TD	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	0	0	6	0	1	0.24	0.2879	0.81	0.0000	3	13	16
1	2	2011-01-01	1	0	1	1	0	0	6	0	1	0.22	0.2727	0.80	0.0000	8	32	40
2	3	2011-01-01	1	0	1	2	0	0	6	0	1	0.22	0.2727	0.80	0.0000	5	27	32
3	4	2011-01-01	1	0	1	3	0	0	6	0	1	0.24	0.2879	0.75	0.0000	3	10	13
4	5	2011-01-01	1	0	1	4	0	0	6	0	1	0.24	0.2879	0.75	0.0000	0	1	1
5	6	2011-01-01	1	0	1	5	0	0	6	0	2	0.24	0.2576	0.75	0.0896	0	1	1

Deloitte.

- We have data with a count of users, and the parameters affecting them

```
In [106]: main_original_df.isnull().sum()

Out[106]: instant       0
          dteday        0
          season        0
          yr            0
          mnth          0
          hr            0
          TD            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
```

No missing values in the original data

Deloitte.

DATA OVERVIEW

Deloitte.

```
[108]: # get columns dtype
       main_original_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 18 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     17379 non-null  int64
 1   dteday      17379 non-null  datetime64[ns]
 2   season      17379 non-null  int64
 3   yr          17379 non-null  int64
 4   mnth        17379 non-null  int64
 5   hr          17379 non-null  int64
 6   TD          17379 non-null  int64
 7   holiday     17379 non-null  int64
 8   weekday     17379 non-null  int64
 9   workingday  17379 non-null  int64
 10  weathersit  17379 non-null  int64
 11  temp        17379 non-null  float64
 12  atemp       17379 non-null  float64
 13  hum         17379 non-null  float64
 14  windspeed   17379 non-null  float64
 15  casual      17379 non-null  int64
 16  registered  17379 non-null  int64
 17  cnt         17379 non-null  int64
dtypes: datetime64[ns](1), float64(4), int64(13)
memory usage: 2.4 MB
```

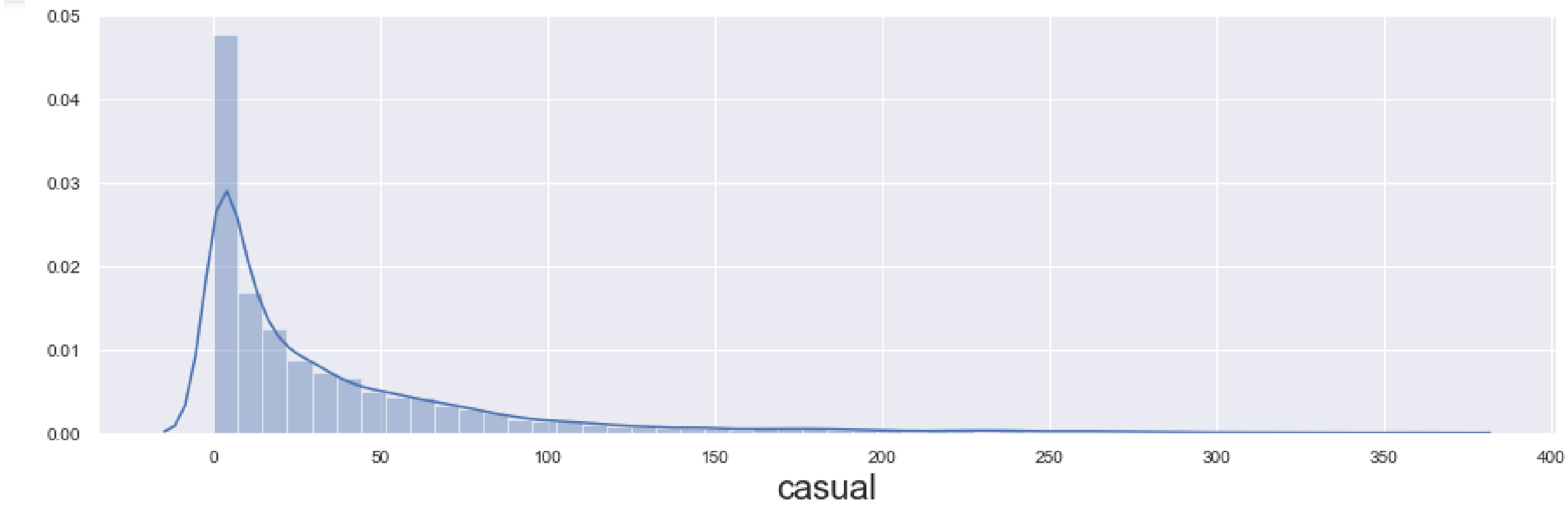
There are 4 dtypes of columns:

- Categorical
- Ordinals
- Continuous
- datetime

Deloitte.

Categorical Ordinals	Continuous	Datetime	Target
• season • yr • hr • TD • holiday • mnth • weekday • workingday • weathersit	• temp • atemp • hum • windspeed	• dteday	• casual • registered • cnt

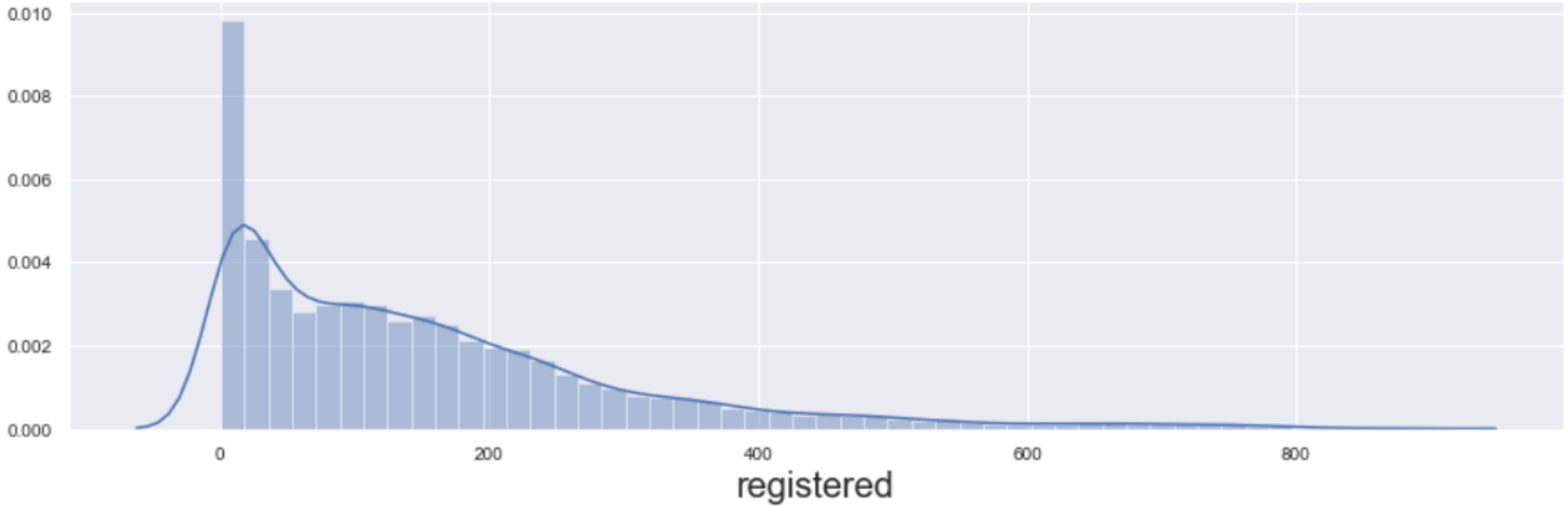
Plot the target - casual users



Deloitte.

A long tail to the right for all targets possible zero-inflated data.

Plot the target - registered users



Deloitte.

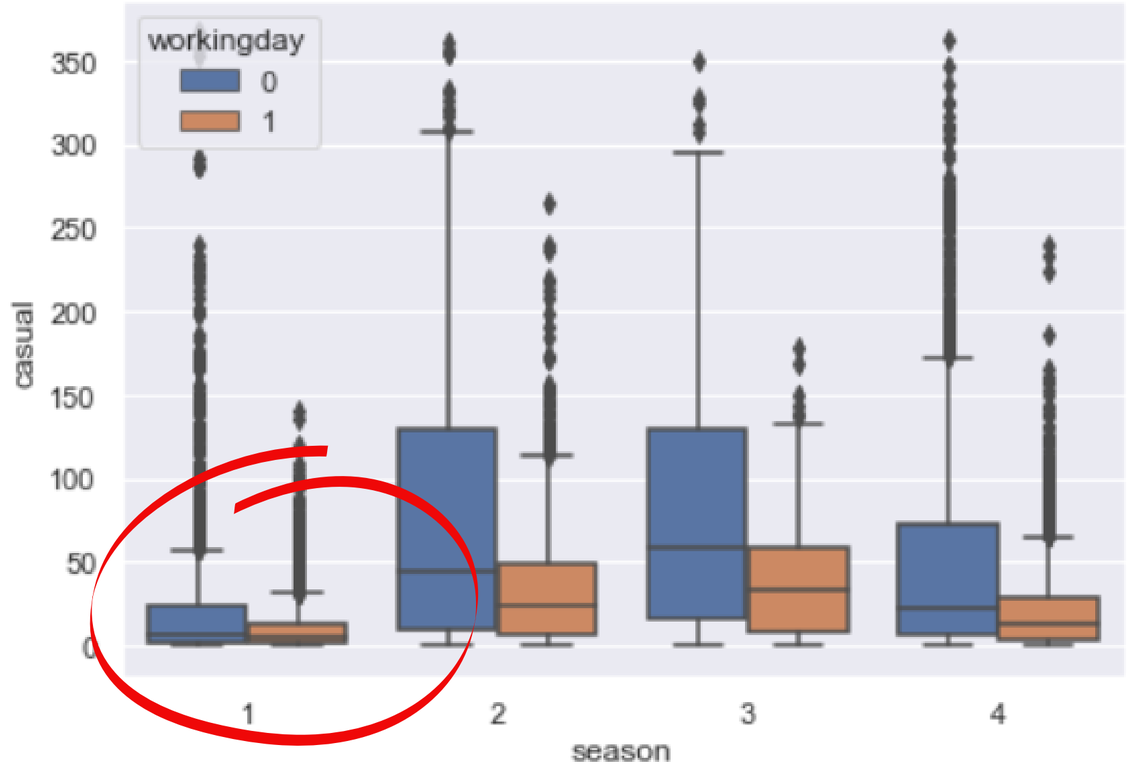
A long tail to the right for all targets possible zero-inflated data.

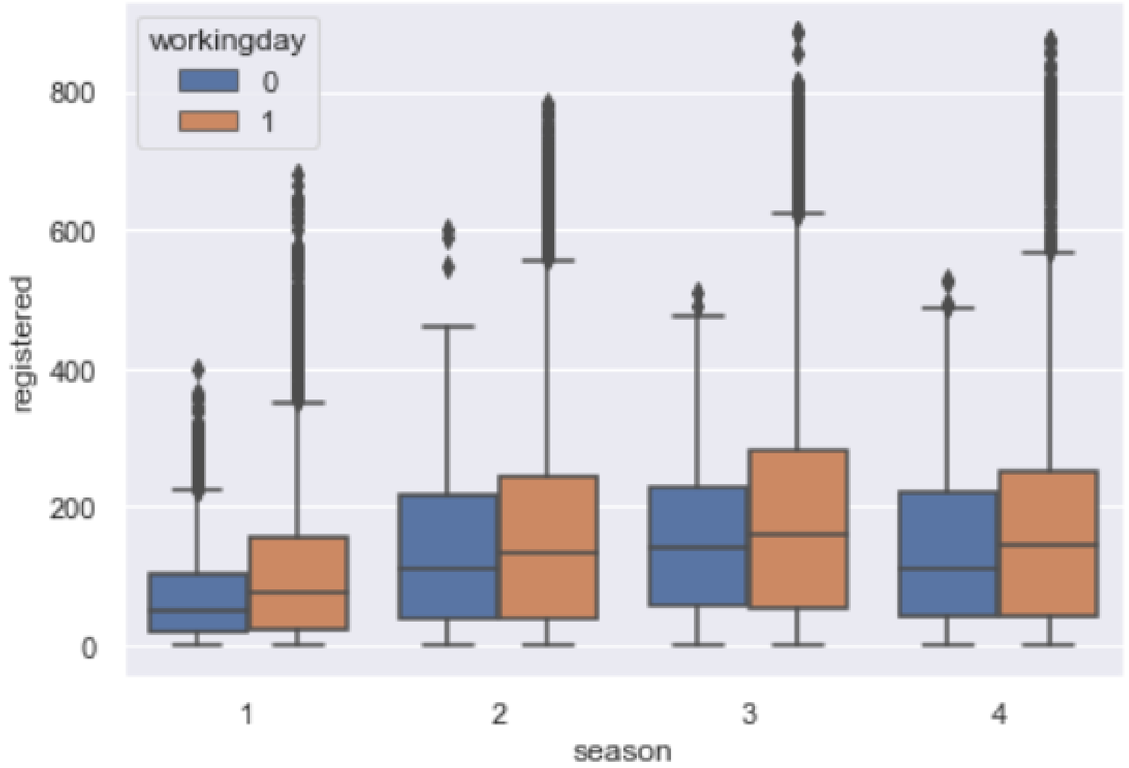
Getting insights using graphs

Deloitte.

Season

Not many casual users in the winter



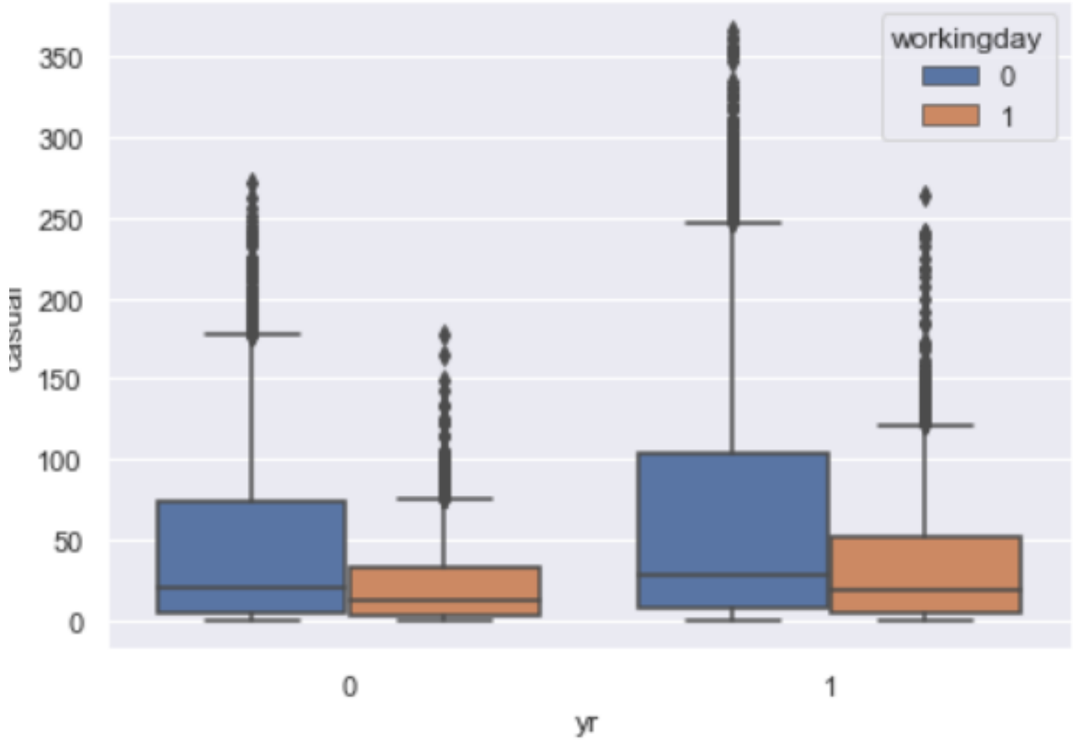


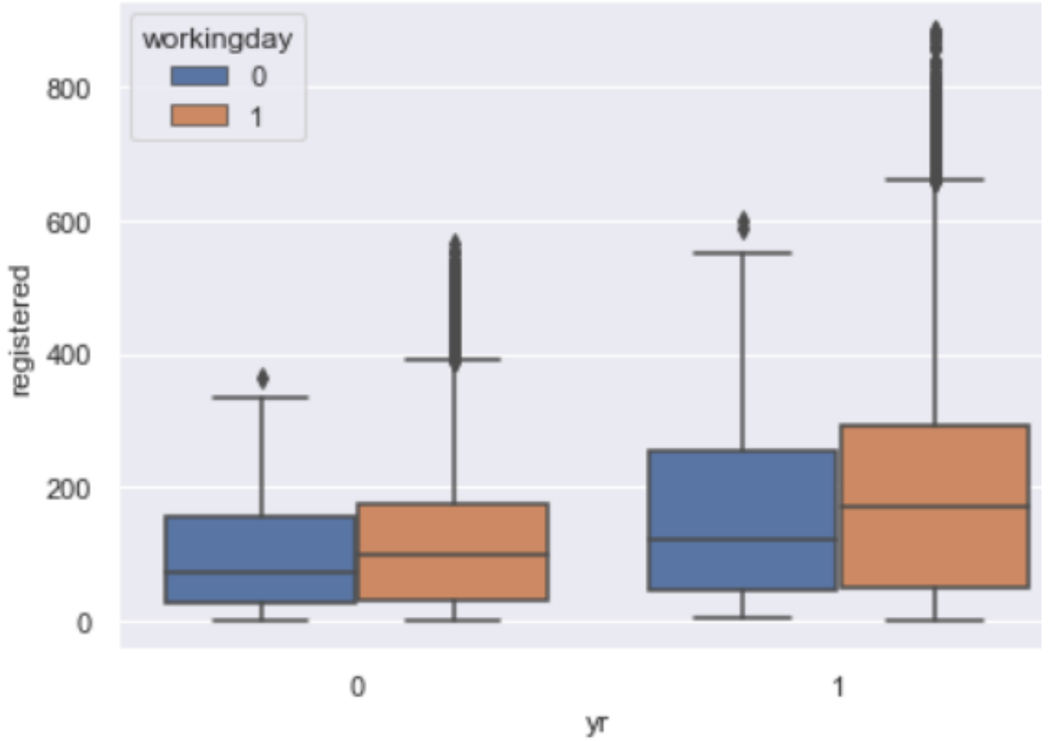
Deloitte.

(1:winter, 2:spring, 3:summer, 4:fall)

Year

there is a rise in users in 2012



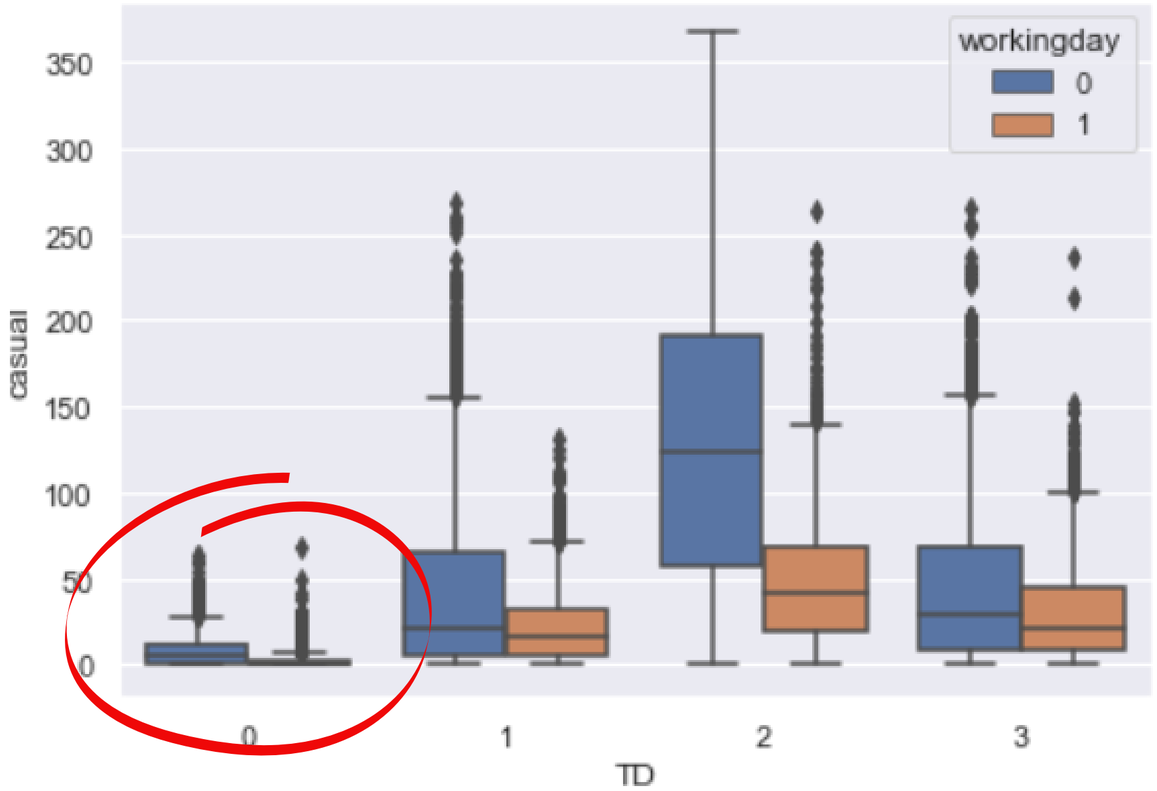


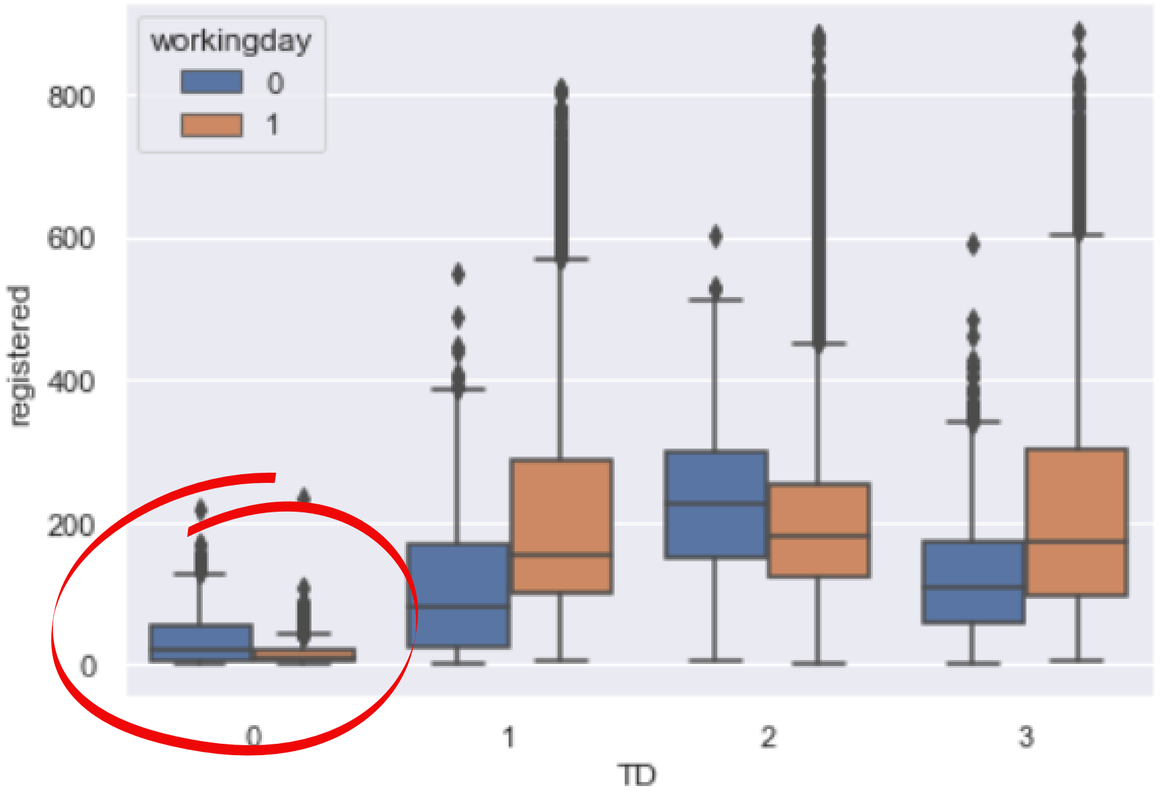
Deloitte.

year (0:2011, 1:2012)

TD

Almost no rental at TD = 0

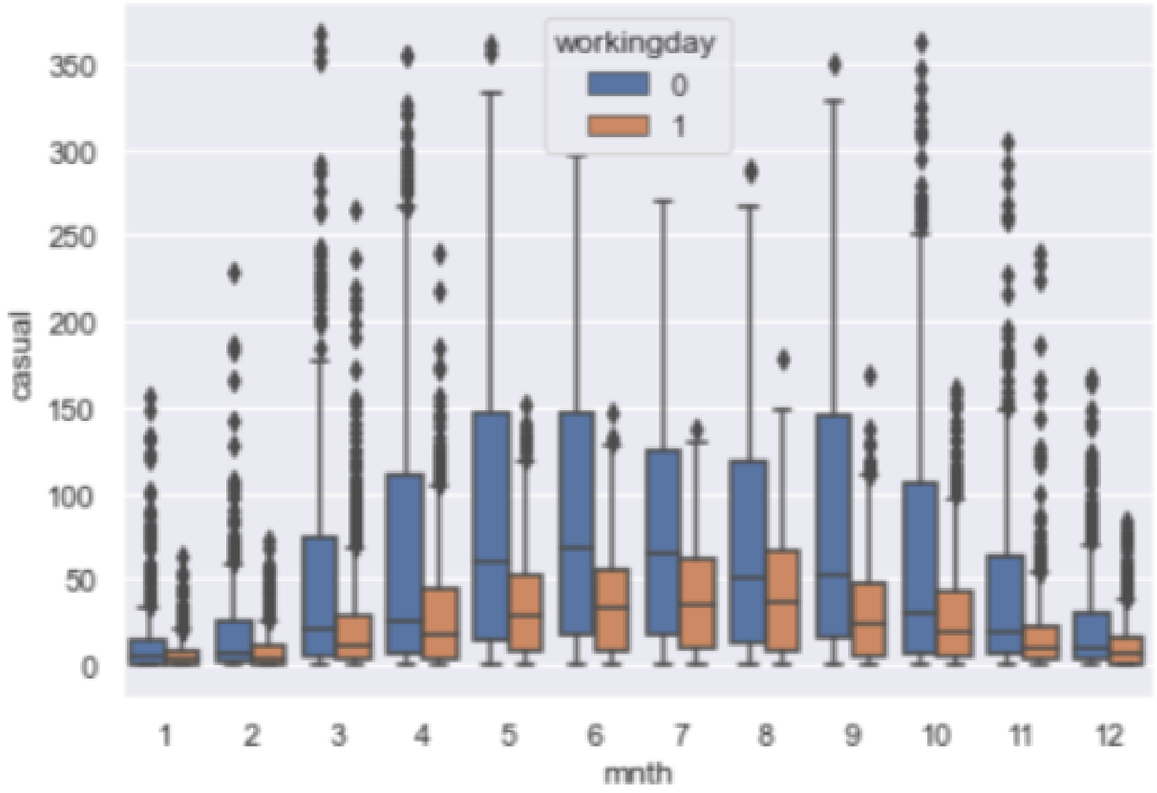


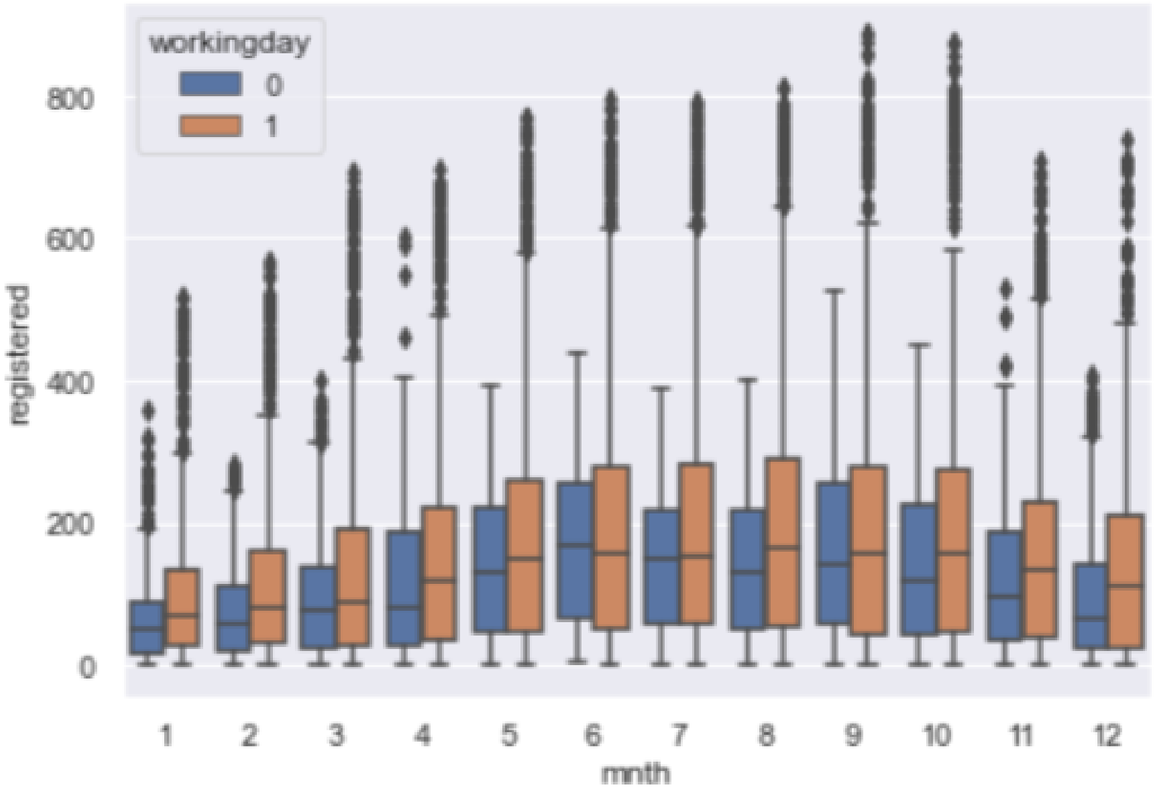


Deloitte.

Month

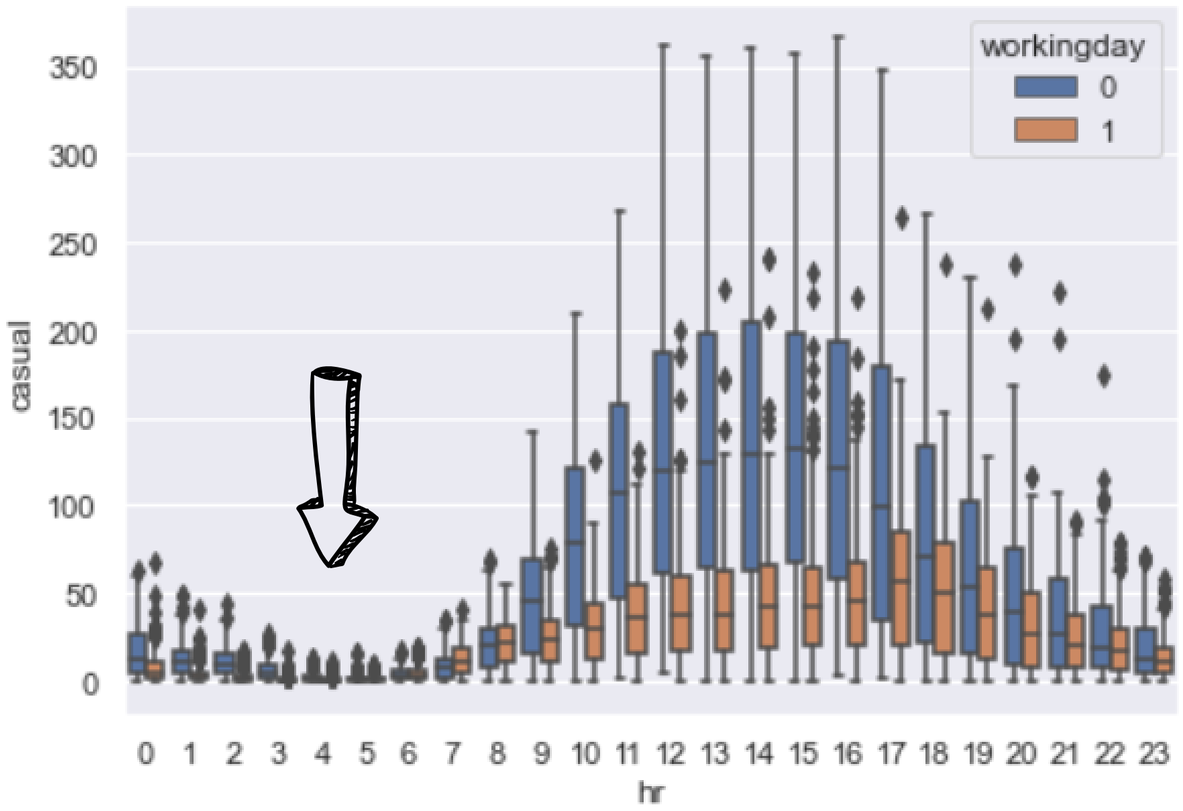
In winter, there is a decline in casual users (months - 11,12,1,2)

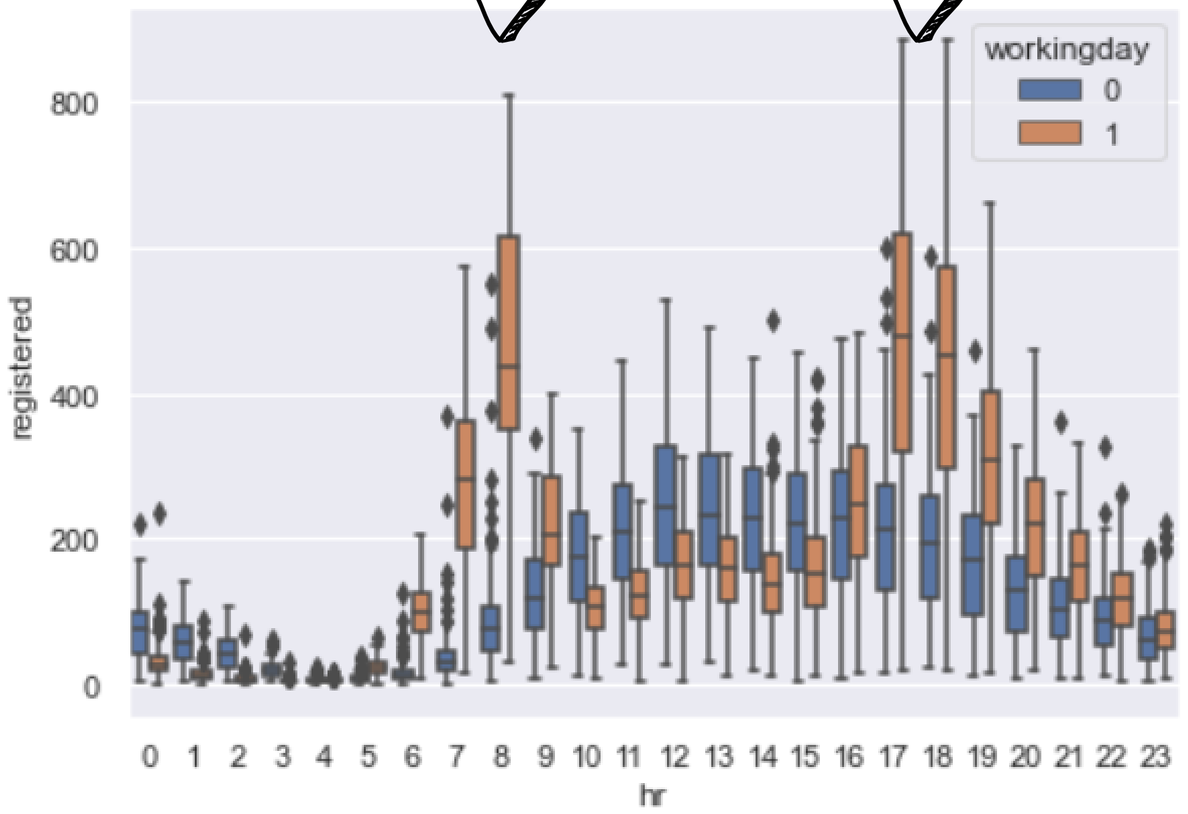




Deloitte.

Hour





Deloitte.

For registered users, there are two peaks in the day.

First peak - between 7-9 (go to work maybe)

Second peak - between 16-19 (return to home maybe)

Both groups (casual and registered) of users rarely used in the night-time (0-6)

More insights..

Deloitte.

Weekday

01

more casual users during the weekends

Holiday

02

registered users have higher median use on holidays, for casual users, the median is the same

Working day

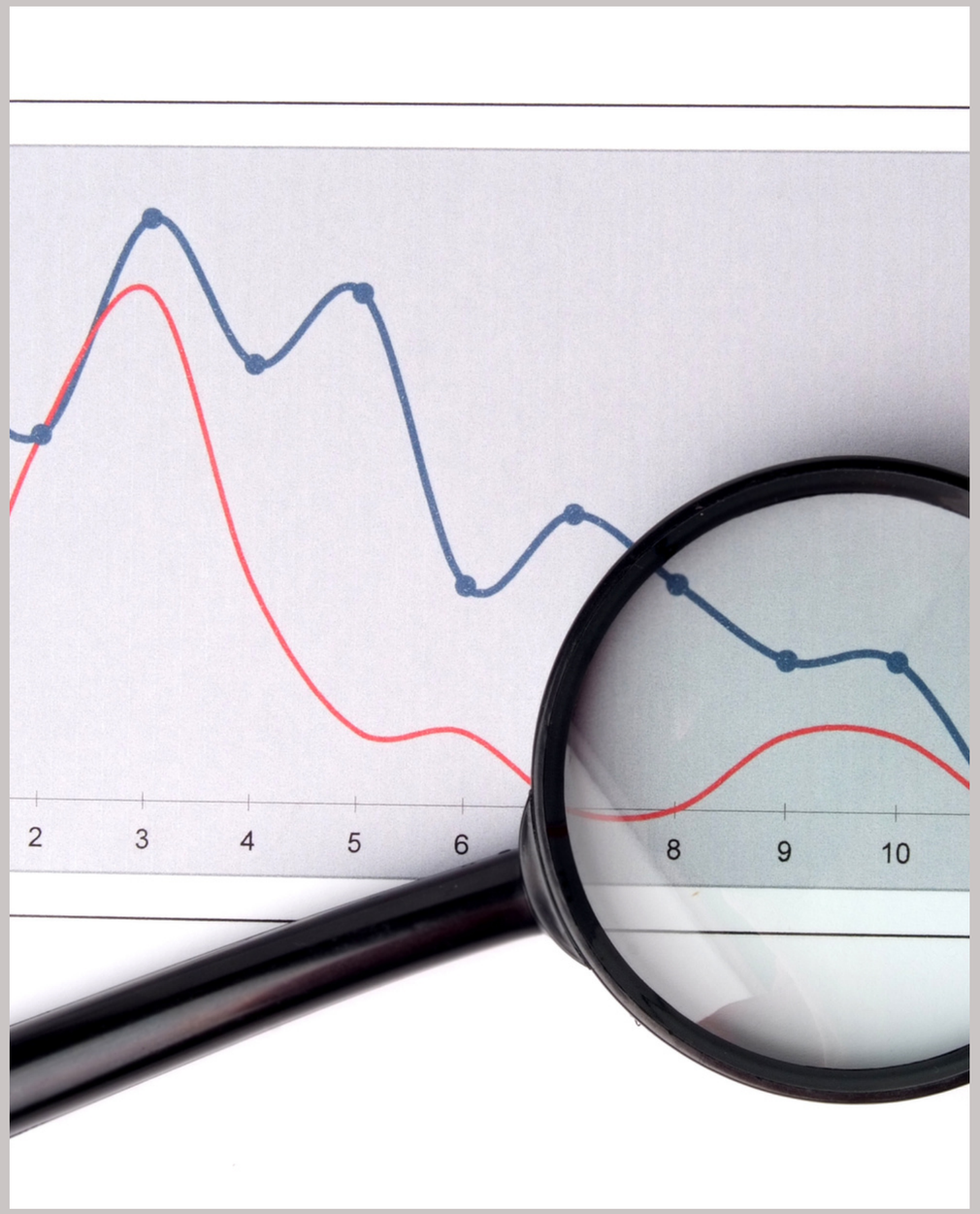
03

again casual users rent more on the weekends or holidays

Weather situation

04

As expected the worst the weather gets, the fewer the users



Deloitte.

correlation

- For continuous data - correlation (Pearson)
- For ordinal categories data - correlation (spearman)
- The ANOVA test for categorical columns (3 groups or more)
- The T-test for categorical columns (binary columns)

Correlation conclusion

- for casual users, the most correlated (spearman, Pearson) columns temp, atemp, and hum, (atemp and temp are basically the same, consider to drop later).

- for registered users, the most correlated (spearman, Pearson) columns are hour, TD, temp, and hum.

- using the ANOVA test we saw that columns -(season, month, weekday, weathersit), $p < 0.05$ for all columns, all variables significantly influence the targets.

- using a t-test we saw that columns - 'yr','holiday','workingday' has significant influence on the target

- (except for the relation of casual ~ workingday with p-value=0 which is >0.05) - has no significant influence!

Deloitte.

	casual	registered	cnt
season	0.120206	0.174226	0.178056
yr	0.142779	0.253684	0.250495
mnth	0.068457	0.122273	0.120638
hr	0.301202	0.374141	0.394071
TD	0.307688	0.405649	0.422126
holiday	0.031564	-0.047345	-0.030927
weekday	0.032721	0.021578	0.026900
workingday	-0.300942	0.134326	0.030284
weathersit	-0.152628	-0.120966	-0.142426
temp	0.459616	0.335361	0.404772
atemp	0.454080	0.332559	0.400929
hum	-0.347028	-0.273933	-0.322911
windspeed	0.090287	0.082321	0.093234
casual	1.000000	0.506618	0.694564
registered	0.506618	1.000000	0.972151
cnt	0.694564	0.972151	1.000000

Deloitte.

correlation (Pearson)

	casual	registered	cnt
season	0.183701	0.182399	0.185179
yr	0.114742	0.221027	0.207500
mnth	0.118084	0.127066	0.125889
hr	0.476666	0.510569	0.510878
TD	0.492870	0.549241	0.546227
holiday	0.015943	-0.043526	-0.029526
weekday	0.012672	0.034711	0.030297
workingday	-0.176671	0.083105	0.021033
weathersit	-0.160280	-0.112062	-0.126276
temp	0.570989	0.373196	0.423330
atemp	0.570419	0.373014	0.423258
hum	-0.388213	-0.338480	-0.359614
windspeed	0.122920	0.122936	0.126629
casual	1.000000	0.781061	0.850525
registered	0.781061	1.000000	0.989444
cnt	0.850525	0.989444	1.000000

Deloitte.

correlation (spearman)

Deloitte.

ANOVA

```
casual ~ season
                sum_sq       df           F        PR(>F)
season    6.104327e+05      1.0  254.771874  6.011151e-57
Residual  4.163524e+07  17377.0         NaN           NaN
---------------------
registered ~ season
                sum_sq       df           F         PR(>F)
season    1.208455e+07      1.0  543.983781  1.685363e-118
Residual  3.860285e+08  17377.0         NaN           NaN
---------------------
casual ~ mnth
                sum_sq       df          F        PR(>F)
mnth      1.979802e+05      1.0  81.819045  1.645466e-19
Residual  4.204769e+07  17377.0        NaN           NaN
---------------------
registered ~ mnth
                sum_sq       df           F        PR(>F)
mnth      5.952061e+06      1.0  263.741046  7.122155e-59
Residual  3.921610e+08  17377.0         NaN           NaN
---------------------
casual ~ weekday
                sum_sq       df         F    PR(>F)
weekday   4.523207e+04      1.0  18.62534  0.000016
Residual  4.220044e+07  17377.0       NaN       NaN
---------------------
registered ~ weekday
                sum_sq       df         F    PR(>F)
weekday   1.853636e+05      1.0  8.094592  0.004445
Residual  3.979277e+08  17377.0       NaN       NaN
---------------------
casual ~ weathersit
                  sum_sq       df           F        PR(>F)
weathersit  9.841245e+05      1.0  414.456811  4.525012e-91
Residual    4.126155e+07  17377.0         NaN           NaN
---------------------
```

Deloitte.

T-test

```
casual ~ yr
(-19.016156613561954, 8.099087744279772e-80, 17377.0)
registered ~ yr
(-34.572089326972, 2.361304684494619e-253, 17377.0)
casual ~ holiday
(-4.162850031219883, 3.158140321233628e-05, 17377.0)
registered ~ holiday
(6.2481682519755175, 4.249584200325769e-10, 17377.0)
casual ~ workingday
(41.59924056323956, 0.0, 17377.0)
registered ~ workingday
(-17.8690171201076, 8.836081675926127e-71, 17377.0)
```

relation of casual ~ working day with p-value=0 which is >0.05- has no significant influence!

Feature Engineering

- **one hot encoding**
- create index
- rush hour
- bad temp/hum

Deloitte.

Create index columns for rush hour and bad temp/hum

- The hour, humidity, and temperature, all have an effect on the count of users.
- I created index columns for specific situations for the model to recognize easier

```python
# create a column for rush hour temp and hum
def rush(row):
    if (row['hr'] in [7,8,9,16,17,18,19]) & (row['workingday'] == 1):
        val = 1
    else:
        val = 0
    return val

def bad_hum(row):
    if (row['hum'] > 0.7) | (row['hum'] <0.2):
        val = 1
    else:
        val = 0
    return val

def bad_temp(row):
    if (row['temp'] > 0.7) | (row['temp'] <0.2):
        val = 1
    else:
        val = 0
    return val
```

Deloitte.

```
x_data = main_original_df[['season', 'yr', 'mnth', 'hr', 'TD', 'holiday',
                           'weekday', 'workingday', 'weathersit', 'temp', 'atemp', 'hum','windspeed']]



# get dummies
data_dummies = pd.get_dummies(data=x_data,columns=['mnth', 'hr','season','mnth', 'hr','TD','weekday'])

# create a column for rush hour
data_dummies['rush_hour'] = x_data.apply(rush, axis=1)

# create a column for bad temp or hum
data_dummies['bad_hum'] = x_data.apply(bad_hum, axis=1)

data_dummies['bad_temp'] = x_data.apply(bad_temp, axis=1)
```

Deloitte.

Final columns names and data frame shapes

```
data_dummies.columns
```

```
Index(['yr', 'holiday', 'workingday', 'weathersit', 'temp', 'atemp', 'hum',
       'windspeed', 'mnth_1', 'mnth_2', 'mnth_3', 'mnth_4', 'mnth_5', 'mnth_6',
       'mnth_7', 'mnth_8', 'mnth_9', 'mnth_10', 'mnth_11', 'mnth_12', 'hr_0',
       'hr_1', 'hr_2', 'hr_3', 'hr_4', 'hr_5', 'hr_6', 'hr_7', 'hr_8', 'hr_9',
       'hr_10', 'hr_11', 'hr_12', 'hr_13', 'hr_14', 'hr_15', 'hr_16', 'hr_17',
       'hr_18', 'hr_19', 'hr_20', 'hr_21', 'hr_22', 'hr_23', 'season_1',
       'season_2', 'season_3', 'season_4', 'mnth_1', 'mnth_2', 'mnth_3',
       'mnth_4', 'mnth_5', 'mnth_6', 'mnth_7', 'mnth_8', 'mnth_9', 'mnth_10',
       'mnth_11', 'mnth_12', 'hr_0', 'hr_1', 'hr_2', 'hr_3', 'hr_4', 'hr_5',
       'hr_6', 'hr_7', 'hr_8', 'hr_9', 'hr_10', 'hr_11', 'hr_12', 'hr_13',
       'hr_14', 'hr_15', 'hr_16', 'hr_17', 'hr_18', 'hr_19', 'hr_20', 'hr_21',
       'hr_22', 'hr_23', 'TD_0', 'TD_1', 'TD_2', 'TD_3', 'weekday_0',
       'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4', 'weekday_5',
       'weekday_6', 'rush_hour', 'bad_hum', 'bad_temp'],
      dtype='object')
```

```
print(x_casual_train.shape)
print(y_casual_train.shape)
print(x_casual_test.shape)
print(y_casual_test.shape)
```

```
(11643, 98)
(11643,)
(5736, 98)
(5736,)
```

Deloitte.

Build & Evaluate the Model

Deloitte.

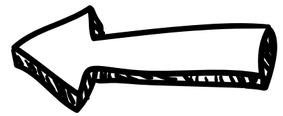
Build the models

- Because both casual and registered users have different relations with variables.
- And cnt is basically the summation of them.
- I decided to build a model for each one (casual and registered)
- If we want after, it's possible to combine the results for the prediction of the total amount (cnt)

Deloitte.

train-test split

```
# y target columns for each model
y_casual_data = main_original_df['casual']
y_registered_data = main_original_df['registered']

# split the data into train test - for casual
x_casual_train, x_casual_test, y_casual_train, y_casual_test = train_test_split(
    data_dummies, y_casual_data, test_size=0.33, random_state=42)

# split the data into train test - for registered
x_registered_train, x_registered_test, y_registered_train, y_registered_test = train_test_split(
    data_dummies, y_registered_data, test_size=0.33, random_state=42)
```



Deloitte.

Define the Models

```
zir = ZeroInflatedRegressor(
    classifier=RandomForestClassifier(random_state=0),
    regressor=RandomForestRegressor(random_state=0)
)

zir2 = ZeroInflatedRegressor(
    classifier=SVC(),
    regressor=LinearRegression()
)

rf = RandomForestRegressor()
lr = LinearRegression()
```

- Use regression models.
- parametric type (linear regression)
- non-parametric type (random forest)
- zero-inflated type.

Using cross-validation to evaluate models performance

Deloitte.

Casual Users Evaluations

```
print('casual for lr')
print(np.abs(np.mean(cross_val_score(lr, x_casual_train, y_casual_train, cv=5,scoring='r2'))))
print(np.abs(np.mean(cross_val_score(lr, x_casual_train, y_casual_train, cv=5,scoring='neg_root_mean_squared_error'))))
print('----------------------')

print('casual for zr')
print(np.abs(np.mean(cross_val_score(zir, x_casual_train, y_casual_train, cv=5,scoring='r2'))))
print(np.abs(np.mean(cross_val_score(zir, x_casual_train, y_casual_train, cv=5,scoring='neg_root_mean_squared_error'))))
print('----------------------')

print('casual for rf')
print(np.abs(np.mean(cross_val_score(rf, x_casual_train, y_casual_train, cv=5,scoring='neg_root_mean_squared_error'))))
print(np.abs(np.mean(cross_val_score(rf, x_casual_train, y_casual_train, cv=5,scoring='r2'))))
print('----------------------')
```

```
casual for lr
0.5927308336732089
31.29383925121557
----------------------
casual for zr
0.8883458939734321
16.3788387303898
----------------------
casual for rf
16.310224040876495
0.887760380920853 7
----------------------
```

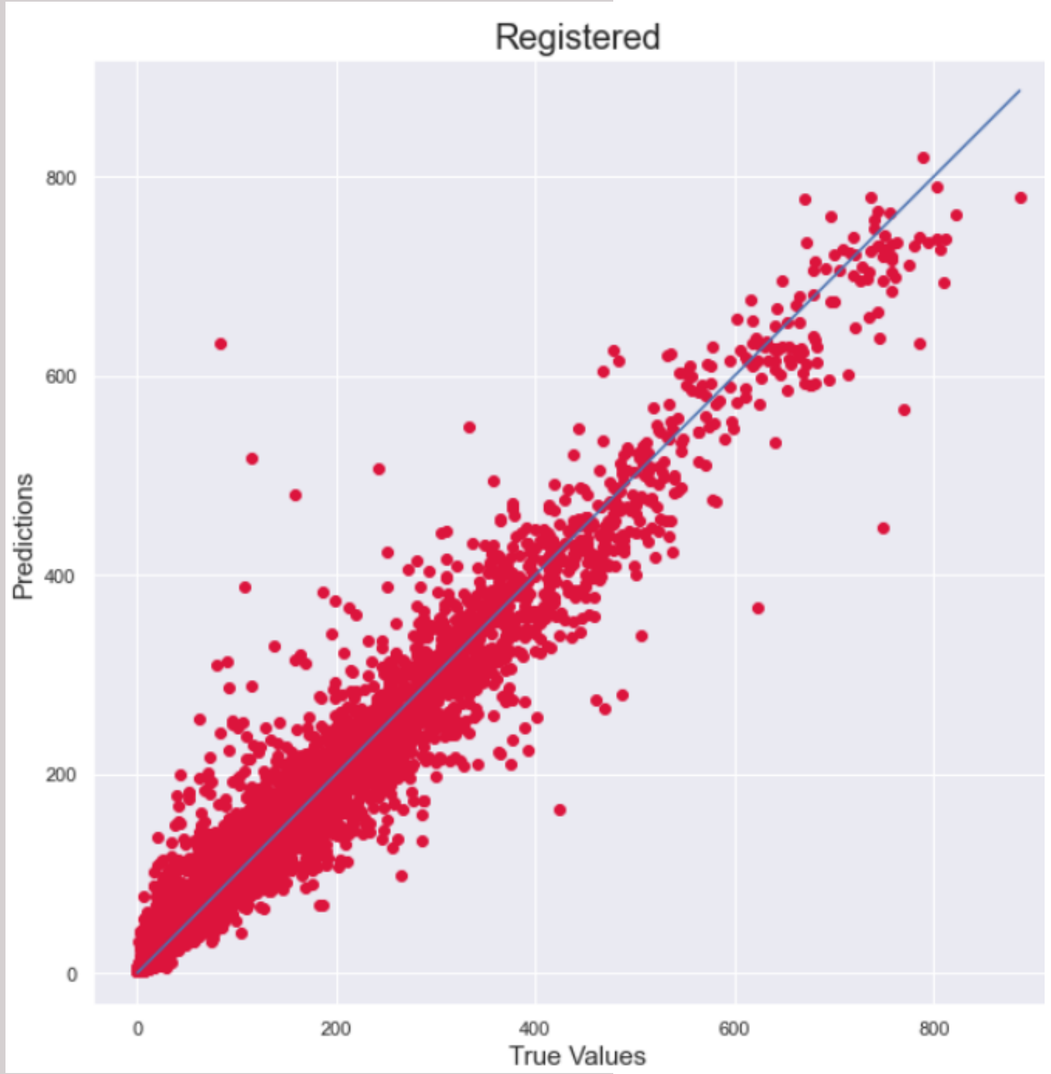
Deloitte.

Registered Users Evaluations

```
print('registered for lr')
print(np.abs(np.mean(cross_val_score(lr,x_registered_train, y_registered_train, cv=5,scoring='r2'))))
print(np.abs(np.mean(cross_val_score(lr, x_registered_train, y_registered_train, cv=5,scoring='neg_root_mean_squared_error'))))
print('----------------------')

print('registered for zr')
print(np.abs(np.mean(cross_val_score(zir, x_registered_train, y_registered_train, cv=5,scoring='r2'))))
print(np.abs(np.mean(cross_val_score(zir, x_registered_train, y_registered_train, cv=5,scoring='neg_root_mean_squared_error'))))
print('----------------------')

print('registered for rf')
print(np.abs(np.mean(cross_val_score(rf, x_registered_train, y_registered_train, cv=5,scoring='neg_root_mean_squared_error'))))
print(np.abs(np.mean(cross_val_score(rf, x_registered_train, y_registered_train, cv=5,scoring='r2'))))
print('----------------------')
```

```
registered for lr
0.768143329159261
9
73.51736190434767
----------------------
registered for zr
0.938860786144175
3
37.733904530659
----------------------
registered for rf
38.10148587368103
0.938590442456110
8
----------------------
```

Deloitte.



Deloitte.

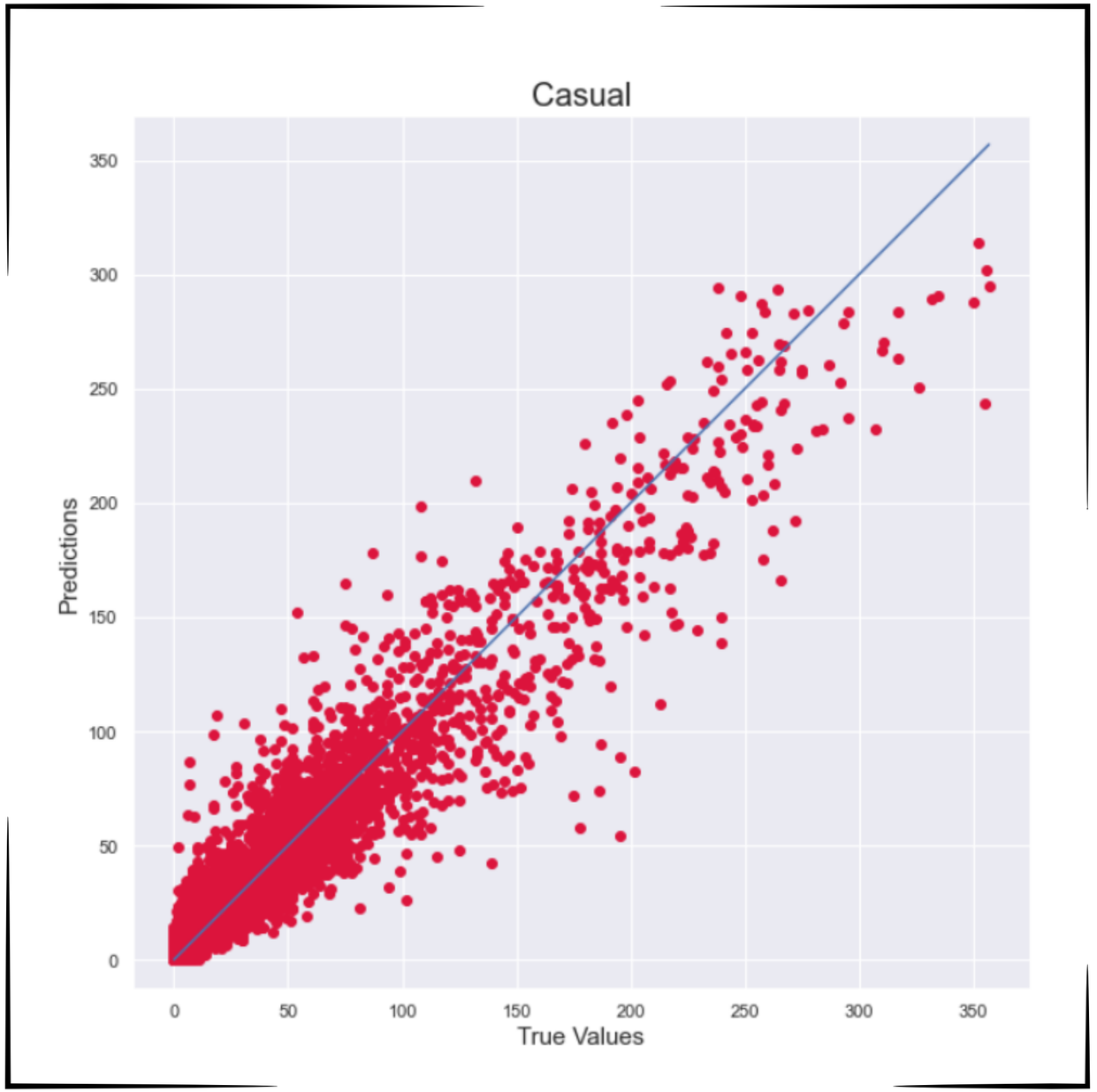
Plot predicted VS true

In [129]:

```
# rf = RandomForestRegressor(n_estimators=500)

true_value = y_registered_test
fited_model = zir.fit(x_registered_train, y_registered_train)
predicted_value = fited_model.predict(x_registered_test)
plt.figure(figsize=(10,10))
plt.scatter(true_value, predicted_value, c='crimson')
# plt.yscale('log')
# plt.xscale('log')

p1 = max(max(predicted_value), max(true_value))
p2 = min(min(predicted_value), min(true_value))
plt.plot([p1, p2], [p1, p2], 'b-')
plt.title('Registered', fontsize=20)
plt.xlabel('True Values', fontsize=15)
plt.ylabel('Predictions', fontsize=15)
plt.axis('equal')
plt.show()
```

Plot predicted VS true

In [130]:
```
# rf = RandomForestRegressor(n_estimators=1000)

true_value = y_casual_test
fited_model = zir.fit(x_casual_train, y_casual_train)
predicted_value = fited_model.predict(x_casual_test)
plt.figure(figsize=(10,10))
plt.scatter(true_value, predicted_value, c='crimson')
# plt.yscale('log')
# plt.xscale('log')

p1 = max(max(predicted_value), max(true_value))
p2 = min(min(predicted_value), min(true_value))
plt.plot([p1, p2], [p1, p2], 'b-')
plt.title('Casual', fontsize=20)
plt.xlabel('True Values', fontsize=15)
```



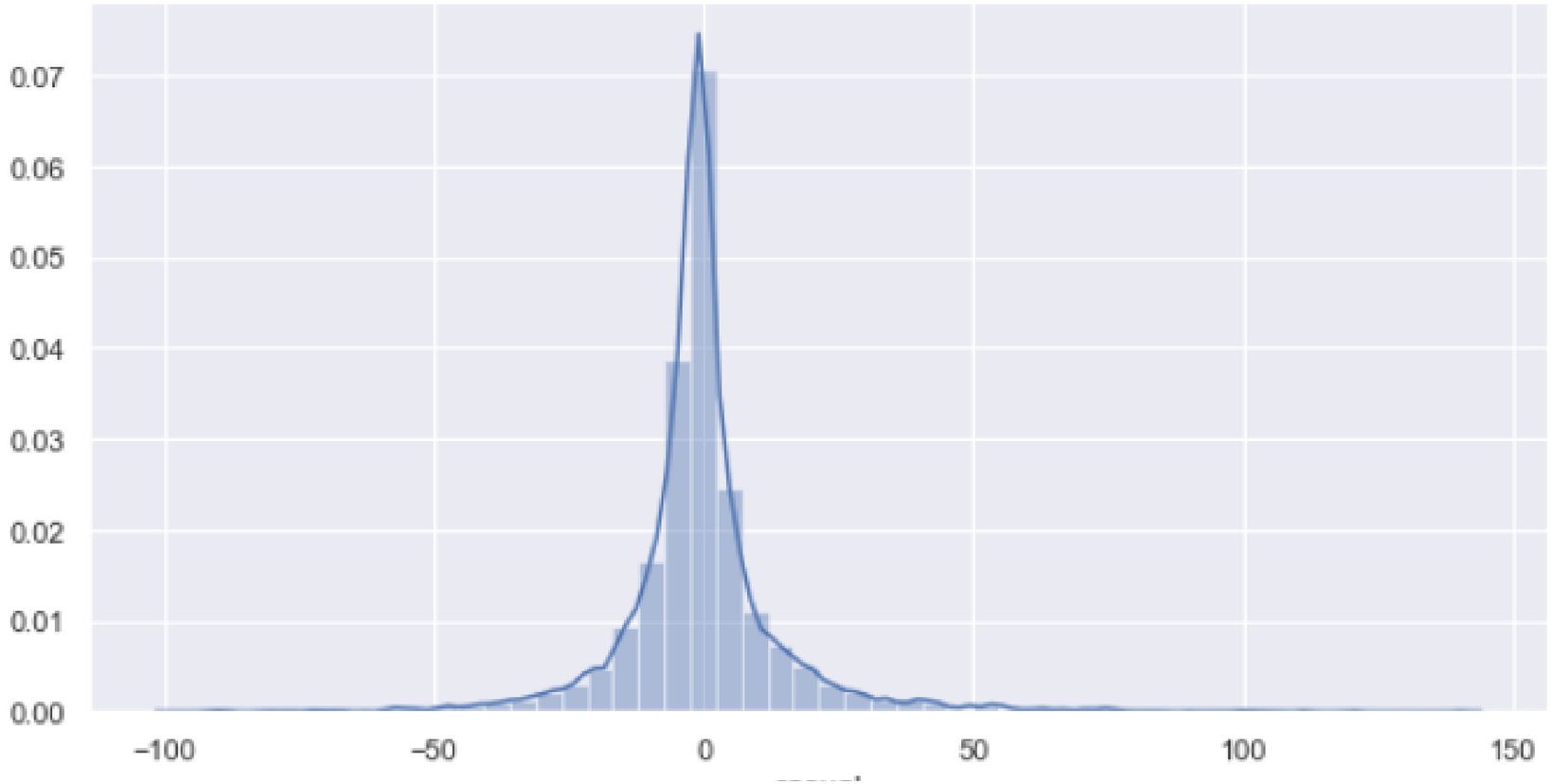
Deloitte.

Plot residuals

Deloitte.

```
fig, ax1 = plt.subplots(1,figsize=(10,5))
sns.distplot((true_value-predicted_value))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bf80625608>
```



Conclusions

Deloitte.

Deloitte.

conclution

- Registered and Casual users have some related characteristics and some different behaviors.
- The main influential parameters are hour, TD, temp, and humidity.
- The Random Forest regressor and ZIR are both best fitted to the data.
- if I had more time, I would focus on the ZIR model, and maybe build the classifier myself (by logic condition)
- Results:

```
registered for lr
0.7681433291592619
73.51736190434767
----------
registered for zr
0.9388607861441753
37.733904530659
----------
registered for rf
38.10148587368103
0.9385904424561108
```

```
casual for lr
0.5927308336732089
31.29383925121557
----------
casual for zr
0.8883458939734321
16.3788387303898
----------
casual for rf
16.310224040876495
0.8877603809208537
----------
```