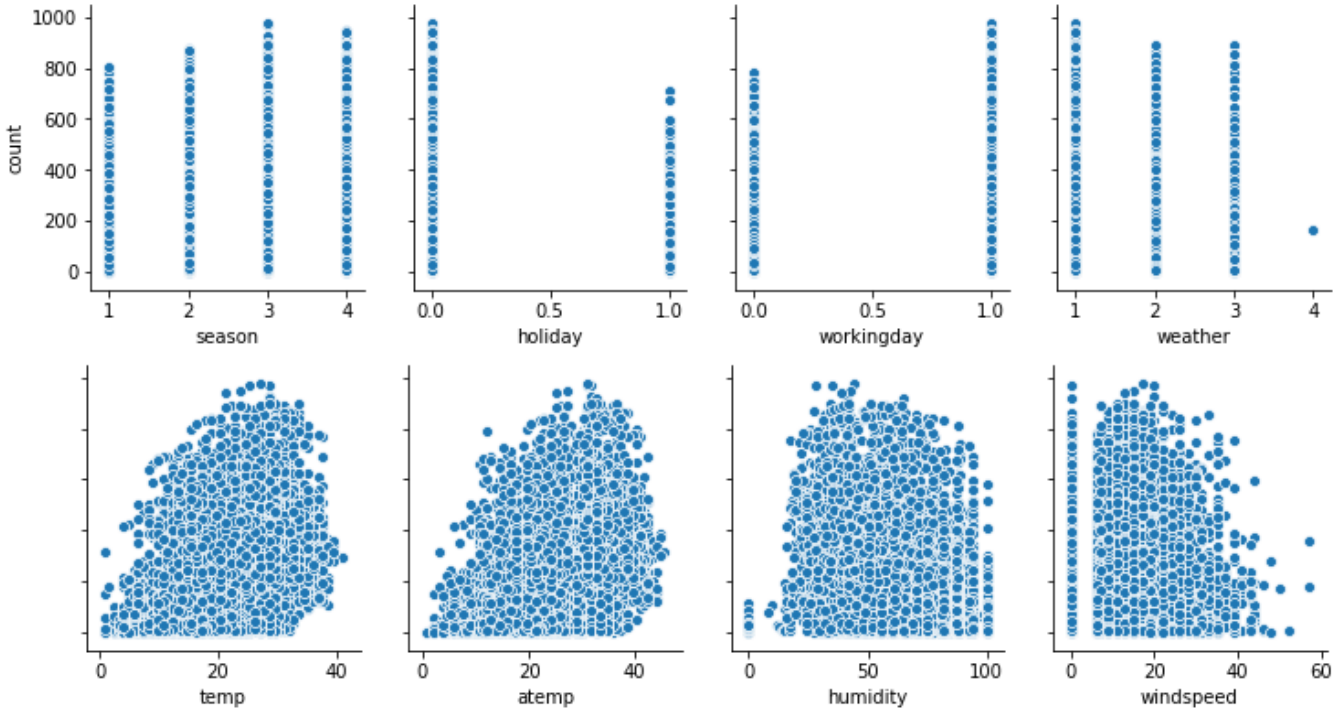
Bike Rental Regressor

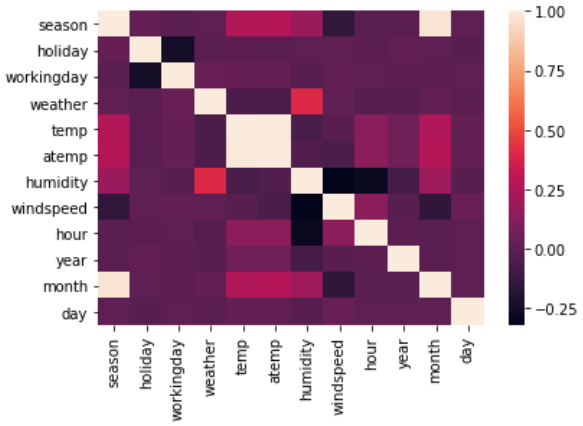
-Zamfir Ovidiu-

1. Input si Dataset

Pentru vizualizarea datelor am folosit pairplot din libraria seaborn, cu ajutorul caruia se poate vedea distributia valorilor ‘count’. Neavand o distributie regulata este greu sa incadram datele intr-o anumita functie. Se observa totusi ca numarul de biciclete inchiriate nu este influentat neaparat de season sau workingday/holiday, cum s-ar crede, ci mai degraba de temp, weather si windspeed. Nu am luat in considerare humidity deoarece este corelat cu weather.



In analiza corelatiilor am utilizat heatmap din seaborn, in care am afisat matricea corelatiilor din X, calculata cu pd.DataFrame().corr().



Cu ajutorul acestui tool se pot observa anumite corelatii intre feature-uri, spre exemplu temp – atemp, weather – humidity, month – season.

2. Preprocesarea datelor

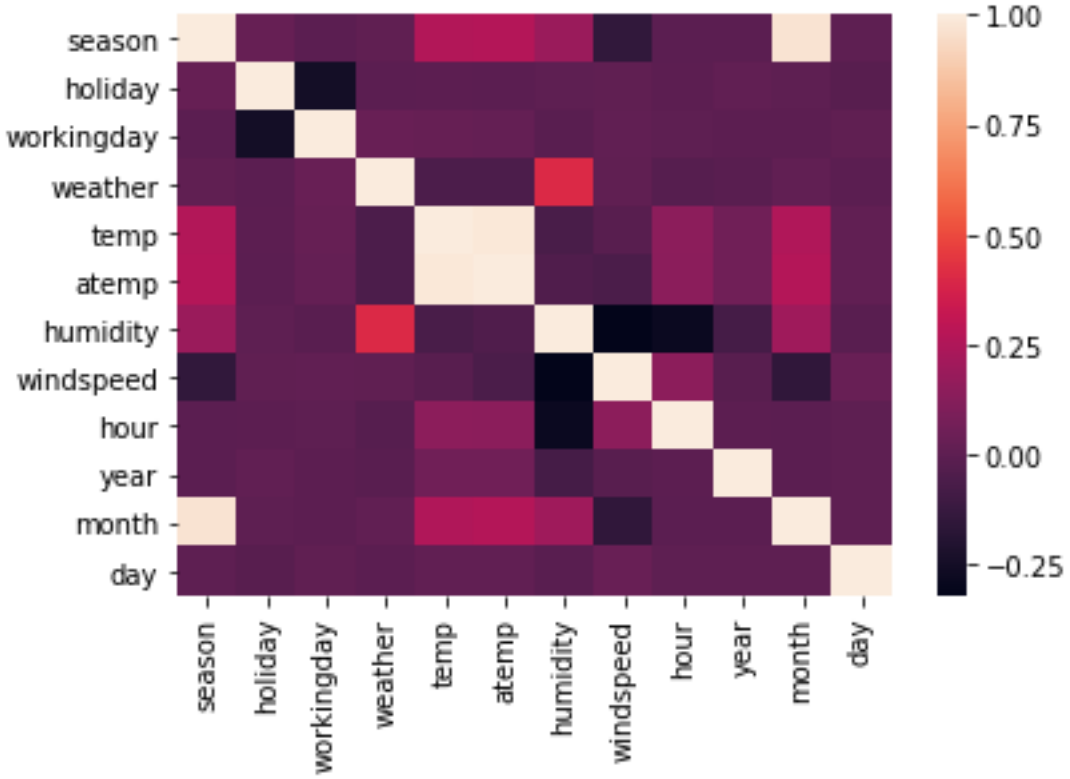
Am impartit variabila ‘datatime’ in 3 coloane diferite : year, month, hour.

Deoarece acest lucru dureaza foarte mult, mai mult decat algoritmul in sine, am rulat operatia de ‘split’ a coloanei ‘datetime’ o singura data si am salvat rezultatele in train2.csv.

La implementarea cu Keras ANN am facut Feature Selection pe baza corelatiilor si am eliminat coloanele season, holiday, humidity si temp, reusind astfel sa obtinem cea mai buna precizie.

Pentru restul implementarilor, nu am obtinut rezultate mai bune cu Feature Selection, asa ca am renuntat la acesta.

Matricea Corelatiilor dintre feature-uri :



Se poate observa ca anumite feature-uri sunt bine corelate intre ele, spre exemplu season-month, temp-atemp, insa se gasesc si corelatii mai putin puternice, cum ar fi weather-humidity.

Am implementat One Hot Encoding pe variabilele weather, year, month si hour.

Datele au fost scalate atat cu MinMaxScaler, cat si cu StandardScaler, insa MinMaxScaler a obtinut rezultate mai bune.

Asadar, pipeline-ul final de preprocesare este urmatorul:

- Split pe datetime
- Feature Selection (optional)
- One Hot Encoding
- MinMax scaling

3. Algoritmii abordati:

Keras ANN:

- Scor 0.27
- Am eliminat feature-urile season, holiday, windspeed, humidity, temp
- 1 input layer + 2 hidden layers + 1 output layer
- Numarul de epoci: 150
- Batch Size: 50

XGBoost:

- Scor: 0.329
- Nu am folosit Feature Selection
- Parametrii: {
- 'booster': 'gbtree',
- 'objective':'reg:linear',
- 'learning_rate': 0.05,
- 'max_depth': 10,
- 'gamma': 0,
- 'min_child_weight': 1,
- 'grow_policy': 'lossguide',
- 'silent': 1,
- 'subsample': 0.7,
- 'colsample_bytree': 0.7,
- 'n_estimators': 100,
- 'tree_method': 'gpu_exact',
- }

GridSearchCV cu XGBoost si PCA:

- Scor : 0.43
- Nu am folosit Feature Selection
- Numarul de componente pentru PCA: 25
- Parametrii: {
- 'booster': ['gbtree'],
- 'objective':['reg:linear'],
- 'learning_rate': [.03, 0.05, .07],
- 'max_depth': [5, 6, 7, 10, 30],
- 'min_child_weight': [1, 2, 3, 4],
- 'silent': [1],
- 'subsample': [0.7],
- 'colsample_bytree': [0.7],
- 'n_estimators': [100, 500],
- }

Keras ANN cu PCA:

- Scor: 0.38
- 1 input layer + 2 hidden layers + 1 output layer
- Numarul de componente pentru PCA: 25
- Nu am folosit Feature Selection
- Numarul de epoci : 150
- Batch Size: 50

GridSearchCV cu AdaBoost, PCA si DecisionTree:

- Scor: 0.38
- Nu am folosit Feature Selection
- Parametrii Grid Search:
- {
- "base_estimator": [DecisionTreeRegressor()],
- "learning_rate": [0.01, 0.05, 0.1, 0.2, 0.5, 1.0, 1.5, 2.0],
- "n_estimators": [50, 100, 150],
- 'loss': ['linear', 'square', 'exponential'],
- "random_state": [0]
- }

Metrica folosita este RMSLE(Root Mean Squared Logarithmic Error), calculata in felul urmator:

```
31 # Metrica RMLSE
32 def RMSLE(Y_test, Y_predict):
33     return tf.math.sqrt(tf.math.reduce_mean(tf.math.square(tf.math.log1p(Y_test)
34                                                            - tf.math.log1p(Y_predict))))
```

4. Concluzii finale

Cel mai bun algoritm folosit este Artificial Neural Network din libraria Keras, obtinand o precizie de 0.27 folosind metrica RMSLE(Root Mean Squared Logarithmic Error).

Acest algoritm foloseste 4 layere de neuroni:

- Primul layer de input cu 41 de neuroni si functie de activare ‘relu’
- Doua hidden layers cu 21, respectiv 11 neuroni ce folosesc tot ‘relu’
- Un output layer cu 1 singur neuron, cu functia de activare ‘linear’

Prin parametrul ‘validation_data’ al modelului putem testa algoritmul pe setul de test dupa fiecare epoca, fara a il antrena pe acesta, algoritmul antrenandu-se doar pe setul de train. La finalul fiecarei epoci vom avea 2 rezultate: RMSLE si val_RMSLE, ultimul reprezentand precizia pe setul de test.