zulonas / **lab32.ipynb**

Created 16 months ago

Star

Code Revisions 2

lab32.ipynb

lab32.ipynb

Raw

Open in Colab

In [196…

```
import math
import keras
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
```

1. Pasirinkite tikslo atributą iš 1 laboratorinio darbo duomenų rinkinio (jei tikslo atributas nebuvo apibrėžtas). (Pastaba: pavyzdžiui, banko klientų duomenų rinkinyje tikslo atributu gali būti laikomi kliento mokumo lygis arba kredito reitingas, filmų duomenų rinkinyje tikslo atributu gali būti sugeneruotas pelnas).
2. Jei reikia, atlikite tikslinių atributų reikšmių pertvarkymus (pvz., platus skaitinių atributų verčių diapazonas keičiamas mažesniu (kategorinių) intervalų skaičiumi (pvz., prognozuojamų reikšmių diapazoną 1..2000 galima pakeisti 1...5 intervalais).
3. Sukurkite reikšmės prognozavimo ar klasifikacijos modelį. Python mokomoji medžiaga pateikta adresu https://iamtrask.github.io/2015/07/12/basic-python-network/
4. Įvertinkite sukurto modelio vidutinį tikslumo įvertį, taikant 10 intervalų kryžminės patikros metodą.
5. Pritaikykite bet kurią iš priemonių (pavyzdžiai pateikiami žemiau), kad padidintumėte vidutinį tikslumą bent 5 procentais ir pakartokite 4-ą darbo eigos žingsnį:
   - Pertvarkyti duomenų rinkinį
   - Pakeiskite mokymosi greitį
   - Pakeiskite aktyvacijos funkciją
   - Pakeisti dirbtinio neuronų tinklo (DNT) struktūrą

In [197…

```
data = pd.read_csv('https://storage.googleapis.com/mledu-datasets/california_housing_train.csv')
```

In [198…

```
data.describe()
```

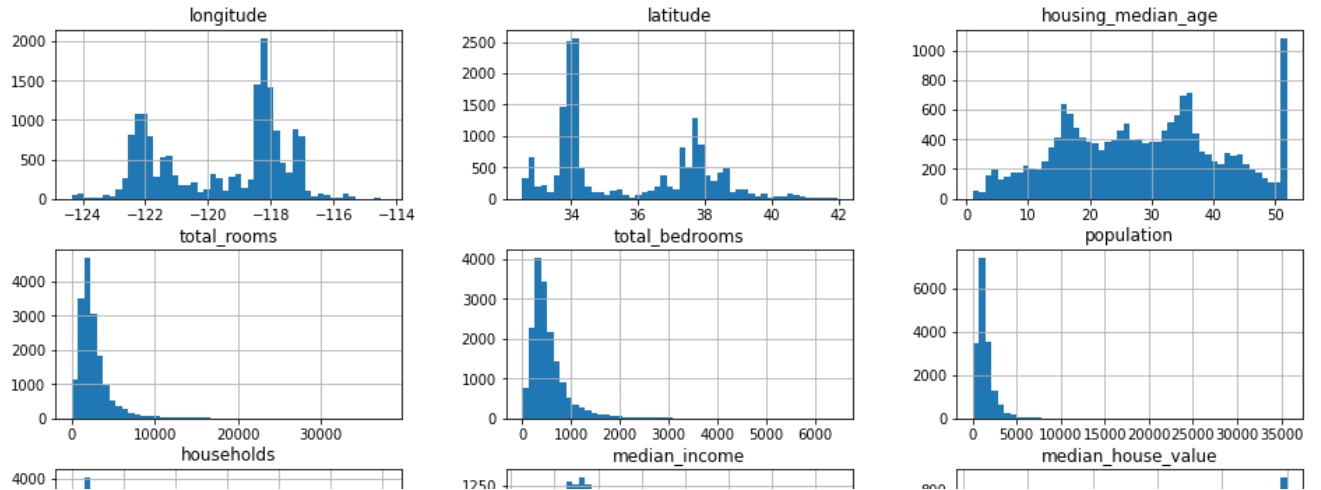
Out[198…

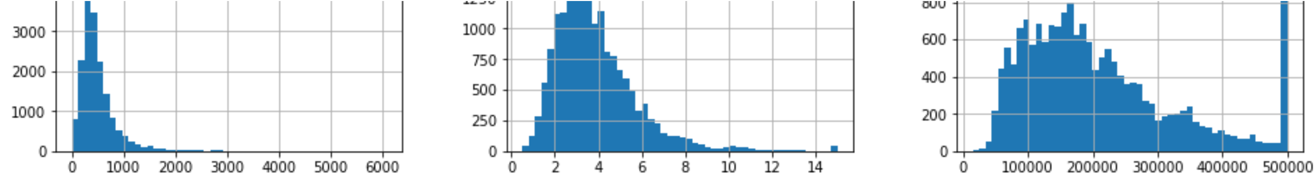
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_incom
count	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.00000
mean	-119.562108	35.625225	28.589353	2643.664412	539.410824	1429.573941	501.221941	3.88357
std	2.005166	2.137340	12.586937	2179.947071	421.499452	1147.852959	384.520841	1.90815
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.49990
25%	-121.790000	33.930000	18.000000	1462.000000	297.000000	790.000000	282.000000	2.56637
50%	-118.490000	34.250000	29.000000	2127.000000	434.000000	1167.000000	409.000000	3.54460
75%	-118.000000	37.720000	37.000000	3151.250000	648.250000	1721.000000	605.250000	4.76700
max	-114.310000	41.950000	52.000000	37937.000000	6445.000000	35682.000000	6082.000000	15.00010

In [199…

```
plt.figure()
data.hist(bins=50, figsize=(16,8))
plt.show()
```

<Figure size 432x288 with 0 Axes>





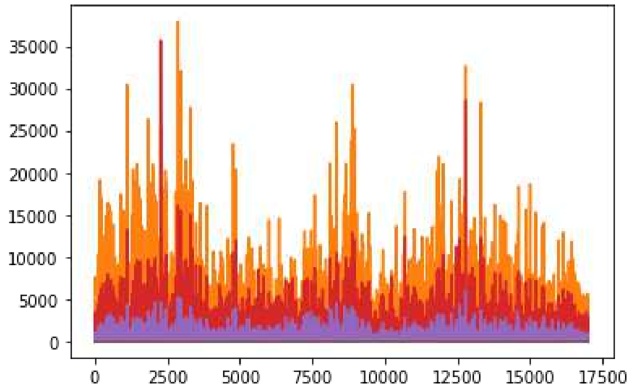
In [200…

```
X = data.drop(columns=['median_house_value', 'longitude', 'latitude'])
y = data['median_house_value']

X = np.array(X)
y = np.array(y)
```

In [201…

```
plt.figure()
plt.plot(X)
plt.show()
```

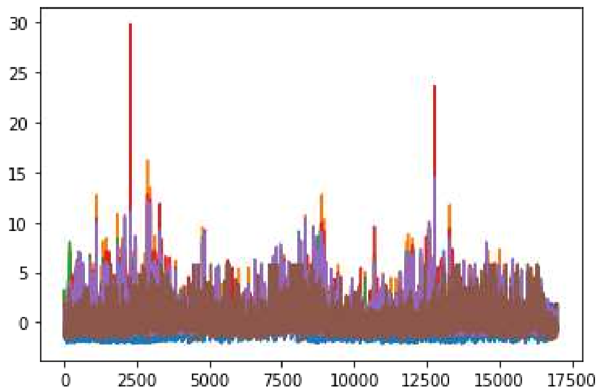


In [202…

```
# Standartization
X_std = np.copy(X)

for i in range(0, 6):
  X_std[:, i] = (X_std[:, i] - X_std[:, i].mean()) / X_std[:, i].std()

plt.figure()
plt.plot(X_std)
plt.show()
```



In [203…

```
X_train, X_test, y_train, y_test = train_test_split(X_std, y, test_size=0.3, random_state=42)
```

In [204…

```
def make_classifier():
    classifier = Sequential()
    classifier.add(Dense(3, kernel_initializer = 'normal', activation = 'relu', input_dim = 6))
    classifier.add(Dense(1, kernel_initializer = 'normal'))
    classifier.compile(optimizer= 'adam', loss = 'mean_squared_error')
    return classifier

make_classifier().summary()
print(make_classifier().get_weights())
```

```
Model: "sequential_164"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_328 (Dense)            (None, 3)                 21
_________________________________________________________________
dense_329 (Dense)            (None, 1)                 4
=================================================================
```

```
Total params: 25
Trainable params: 25
Non-trainable params: 0
_________________________________________________________________
[array([[-0.08211987,  0.09985056, -0.00533692],
       [ 0.08427793,  0.02890524,  0.05158813],
       [ 0.00211019, -0.034972  , -0.04090815],
       [ 0.11112738,  0.00880713,  0.06004889],
       [ 0.00412849, -0.01547211,  0.03698185],
       [ 0.00387961,  0.01784034, -0.04248256]], dtype=float32), array([0., 0., 0.], dtype=float32), ar
ray([[-0.04004624],
       [ 0.00190758],
       [-0.05348462]], dtype=float32), array([0.], dtype=float32)]
```

In [205…

```
regressor = KerasRegressor(build_fn=make_classifier, nb_epoch=1000, batch_size=10)
regressor.fit(X_train, y_train)
```

```
1190/1190 [==============================] - 1s 891us/step - loss: 54815568071.4694
```

Out[205…

```
<tensorflow.python.keras.callbacks.History at 0x7f29f16f5dd0>
```

In [206…

```
make_classifier().get_weights()
```

Out[206…

```
[array([[-0.09521233, -0.05778117,  0.03947655],
        [ 0.04262883, -0.06259254,  0.00380649],
        [-0.03634699,  0.06141048, -0.09260476],
        [-0.15431689,  0.02168981,  0.04822403],
        [-0.01684596, -0.0313106 , -0.02027882],
        [ 0.01559381,  0.0252065 , -0.06454287]], dtype=float32),
 array([0., 0., 0.], dtype=float32),
 array([[-0.02632541],
        [-0.00964384],
        [-0.06122212]], dtype=float32),
 array([0.], dtype=float32)]
```

In [209…

```
y_pred = regressor.predict(X_test)

def get_mse(initial, predicted):
    err = 0.0
    for i in range(len(initial)):
        err += (predicted[i] - initial[i]) ** 2
    return err / len(predicted)

def get_mad(predicted):
    return np.median(np.absolute(predicted - np.median(predicted)))

mse = get_mse(y_test, y_pred)
mad = get_mad(y_pred)

print("MSE = %f" % (mse))
print("MAD = %f" % (mad))
```

```
MSE = 58202736169.152023
MAD = 5.913820
```

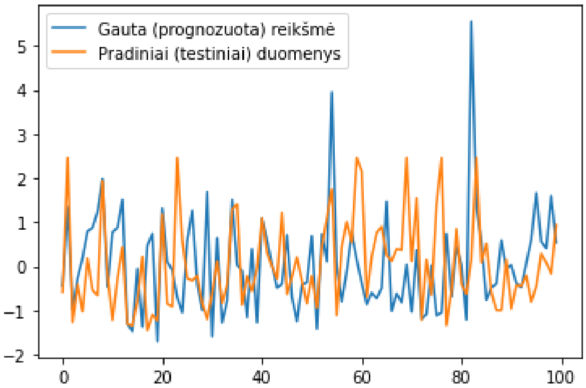
In [213…

```
plt.figure()

y_pred1 = (y_pred - y_pred.mean()) / y_pred.std()
y_test1 = (y_test - y_test.mean()) / y_test.std()

plt.plot(y_pred1[:100], label="Gauta (prognozuota) reikšmė")
plt.plot(y_test1[:100], label="Pradiniai (testiniai) duomenys")

plt.legend()
plt.show()
```



In [187…

```
estimator = KerasRegressor(build_fn=make_classifier, epochs=10, batch_size=10, verbose=1)
```

```
kfold = KFold(n_splits=10)
results = cross_val_score(estimator, X_train, y_train, cv=kfold)
print("Results: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

```
Epoch 1/10
1071/1071 [==============================] - 1s 955us/step - loss: 54996066670.8060
Epoch 2/10
1071/1071 [==============================] - 1s 932us/step - loss: 55662186060.4179
Epoch 3/10
1071/1071 [==============================] - 1s 953us/step - loss: 56319307619.3433
Epoch 4/10
1071/1071 [==============================] - 1s 946us/step - loss: 54847246836.5373
Epoch 5/10
1071/1071 [==============================] - 1s 996us/step - loss: 55714457941.9701
Epoch 6/10
1071/1071 [==============================] - 1s 1ms/step - loss: 55719915095.8806
Epoch 7/10
1071/1071 [==============================] - 1s 979us/step - loss: 54957287129.7910
Epoch 8/10
1071/1071 [==============================] - 1s 930us/step - loss: 54835031259.7015
Epoch 9/10
1071/1071 [==============================] - 1s 930us/step - loss: 55282646806.9254
Epoch 10/10
1071/1071 [==============================] - 1s 924us/step - loss: 55838272114.6269
119/119 [==============================] - 0s 832us/step - loss: 54832996352.0000
Epoch 1/10
1071/1071 [==============================] - 1s 890us/step - loss: 56383452817.1940
Epoch 2/10
1071/1071 [==============================] - 1s 931us/step - loss: 55844543029.4925
Epoch 3/10
1071/1071 [==============================] - 1s 929us/step - loss: 55364479281.6716
Epoch 4/10
1071/1071 [==============================] - 1s 933us/step - loss: 55216567808.0000
Epoch 5/10
1071/1071 [==============================] - 1s 930us/step - loss: 55387290834.1493
Epoch 6/10
1071/1071 [==============================] - 1s 936us/step - loss: 54662843965.1343
Epoch 7/10
1071/1071 [==============================] - 1s 914us/step - loss: 55759364699.7015
Epoch 8/10
1071/1071 [==============================] - 1s 927us/step - loss: 54894335438.3284
Epoch 9/10
1071/1071 [==============================] - 1s 898us/step - loss: 56241406922.5075
Epoch 10/10
1071/1071 [==============================] - 1s 911us/step - loss: 55466686819.3433
119/119 [==============================] - 0s 768us/step - loss: 55414632448.0000
Epoch 1/10
1071/1071 [==============================] - 1s 864us/step - loss: 55718542301.6119
Epoch 2/10
1071/1071 [==============================] - 1s 920us/step - loss: 54813743031.4030
Epoch 3/10
1071/1071 [==============================] - 1s 969us/step - loss: 54522572566.9254
Epoch 4/10
1071/1071 [==============================] - 1s 916us/step - loss: 55730919485.1343
Epoch 5/10
1071/1071 [==============================] - 1s 892us/step - loss: 55138253082.7463
Epoch 6/10
1071/1071 [==============================] - 1s 940us/step - loss: 55309142000.7164
Epoch 7/10
1071/1071 [==============================] - 1s 921us/step - loss: 54806629620.5373
Epoch 8/10
1071/1071 [==============================] - 1s 935us/step - loss: 55336922857.0746
Epoch 9/10
1071/1071 [==============================] - 1s 966us/step - loss: 54536274653.6119
Epoch 10/10
1071/1071 [==============================] - 1s 948us/step - loss: 57365896256.9552
119/119 [==============================] - 0s 784us/step - loss: 58834325504.0000
Epoch 1/10
1071/1071 [==============================] - 1s 844us/step - loss: 56510689665.9104
Epoch 2/10
1071/1071 [==============================] - 1s 894us/step - loss: 56288758101.9701
Epoch 3/10
1071/1071 [==============================] - 1s 941us/step - loss: 55452443233.4328
Epoch 4/10
1071/1071 [==============================] - 1s 893us/step - loss: 53919713486.3284
Epoch 5/10
1071/1071 [==============================] - 1s 929us/step - loss: 56137328957.1343
Epoch 6/10
1071/1071 [==============================] - 1s 934us/step - loss: 55167042491.2239
Epoch 7/10
1071/1071 [==============================] - 1s 977us/step - loss: 56000412545.9104
Epoch 8/10
1071/1071 [==============================] - 1s 964us/step - loss: 54874972672.0000
Epoch 9/10
1071/1071 [==============================] - 1s 960us/step - loss: 55509724083.5821
Epoch 10/10
```

```
1071/1071 [==============================] - 1s 940us/step - loss: 55686599890.1493
119/119 [==============================] - 0s 1ms/step - loss: 53877567488.0000
Epoch 1/10
1071/1071 [==============================] - 1s 922us/step - loss: 56029327016.1194
Epoch 2/10
1071/1071 [==============================] - 1s 979us/step - loss: 55284720403.1045
Epoch 3/10
1071/1071 [==============================] - 1s 920us/step - loss: 55695865378.3881
Epoch 4/10
1071/1071 [==============================] - 1s 910us/step - loss: 56293840007.6418
Epoch 5/10
1071/1071 [==============================] - 1s 877us/step - loss: 55810322133.9701
Epoch 6/10
1071/1071 [==============================] - 1s 930us/step - loss: 53981781664.4776
Epoch 7/10
1071/1071 [==============================] - 1s 883us/step - loss: 55541290318.3284
Epoch 8/10
1071/1071 [==============================] - 1s 892us/step - loss: 55247506863.7612
Epoch 9/10
1071/1071 [==============================] - 1s 925us/step - loss: 56324808432.7164
Epoch 10/10
1071/1071 [==============================] - 1s 908us/step - loss: 54925540695.8806
119/119 [==============================] - 0s 775us/step - loss: 55943786496.0000
Epoch 1/10
1071/1071 [==============================] - 1s 889us/step - loss: 55897793365.9701
Epoch 2/10
1071/1071 [==============================] - 1s 910us/step - loss: 55111885495.4030
Epoch 3/10
1071/1071 [==============================] - 1s 927us/step - loss: 56654168740.2985
Epoch 4/10
1071/1071 [==============================] - 1s 921us/step - loss: 56941020301.3731
Epoch 5/10
1071/1071 [==============================] - 1s 921us/step - loss: 55354673576.1194
Epoch 6/10
1071/1071 [==============================] - 1s 932us/step - loss: 55945209431.8806
Epoch 7/10
1071/1071 [==============================] - 1s 903us/step - loss: 56042017199.7612
Epoch 8/10
1071/1071 [==============================] - 1s 934us/step - loss: 55885961093.7313
Epoch 9/10
1071/1071 [==============================] - 1s 892us/step - loss: 56338904222.5672
Epoch 10/10
1071/1071 [==============================] - 1s 978us/step - loss: 55512477799.1642
119/119 [==============================] - 0s 826us/step - loss: 53468233728.0000
Epoch 1/10
1071/1071 [==============================] - 1s 906us/step - loss: 55690470587.2239
Epoch 2/10
1071/1071 [==============================] - 1s 934us/step - loss: 55451933739.9403
Epoch 3/10
1071/1071 [==============================] - 1s 916us/step - loss: 56123108420.7761
Epoch 4/10
1071/1071 [==============================] - 1s 932us/step - loss: 55436661473.4328
Epoch 5/10
1071/1071 [==============================] - 1s 961us/step - loss: 54774476643.3433
Epoch 6/10
1071/1071 [==============================] - 1s 979us/step - loss: 55057844483.8209
Epoch 7/10
1071/1071 [==============================] - 1s 887us/step - loss: 55137744265.5522
Epoch 8/10
1071/1071 [==============================] - 1s 917us/step - loss: 55291063250.1493
Epoch 9/10
1071/1071 [==============================] - 1s 917us/step - loss: 55581354182.6866
Epoch 10/10
1071/1071 [==============================] - 1s 918us/step - loss: 54540807767.8806
119/119 [==============================] - 0s 813us/step - loss: 56527781888.0000
Epoch 1/10
1071/1071 [==============================] - 1s 879us/step - loss: 55278571229.6119
Epoch 2/10
1071/1071 [==============================] - 1s 873us/step - loss: 56005826796.8955
Epoch 3/10
1071/1071 [==============================] - 1s 954us/step - loss: 54743946366.0896
Epoch 4/10
1071/1071 [==============================] - 1s 910us/step - loss: 55518902331.2239
Epoch 5/10
1071/1071 [==============================] - 1s 918us/step - loss: 55441469887.0448
Epoch 6/10
1071/1071 [==============================] - 1s 909us/step - loss: 55674925950.0896
Epoch 7/10
1071/1071 [==============================] - 1s 923us/step - loss: 55997104213.9701
Epoch 8/10
1071/1071 [==============================] - 1s 902us/step - loss: 55633986659.3433
Epoch 9/10
1071/1071 [==============================] - 1s 902us/step - loss: 55611993126.2090
Epoch 10/10
1071/1071 [==============================] - 1s 991us/step - loss: 55878931941.2537
119/119 [==============================] - 0s 764us/step - loss: 54763802624.0000
Epoch 1/10
```

```
1071/1071 [==============================] - 1s 893us/step - loss: 56632298255.2836
Epoch 2/10
1071/1071 [==============================] - 1s 922us/step - loss: 54929606673.1940
Epoch 3/10
1071/1071 [==============================] - 1s 906us/step - loss: 56004126355.1045
Epoch 4/10
1071/1071 [==============================] - 1s 927us/step - loss: 55149424269.3731
Epoch 5/10
1071/1071 [==============================] - 1s 906us/step - loss: 56291820723.5821
Epoch 6/10
1071/1071 [==============================] - 1s 944us/step - loss: 55978992739.3433
Epoch 7/10
1071/1071 [==============================] - 1s 922us/step - loss: 55650399980.8955
Epoch 8/10
1071/1071 [==============================] - 1s 974us/step - loss: 55391987203.8209
Epoch 9/10
1071/1071 [==============================] - 1s 875us/step - loss: 56141172904.1194
Epoch 10/10
1071/1071 [==============================] - 1s 904us/step - loss: 55651507570.6269
119/119 [==============================] - 0s 951us/step - loss: 55259987968.0000
Epoch 1/10
1071/1071 [==============================] - 1s 860us/step - loss: 54575625578.9851
Epoch 2/10
1071/1071 [==============================] - 1s 875us/step - loss: 56618096143.2836
Epoch 3/10
1071/1071 [==============================] - 1s 917us/step - loss: 55191909395.1045
Epoch 4/10
1071/1071 [==============================] - 1s 887us/step - loss: 55873949638.6866
Epoch 5/10
1071/1071 [==============================] - 1s 935us/step - loss: 55559995254.4478
Epoch 6/10
1071/1071 [==============================] - 1s 926us/step - loss: 55969407919.7612
Epoch 7/10
1071/1071 [==============================] - 1s 912us/step - loss: 56927552294.2090
Epoch 8/10
1071/1071 [==============================] - 1s 954us/step - loss: 55602854403.8209
Epoch 9/10
1071/1071 [==============================] - 1s 940us/step - loss: 54852669795.3433
Epoch 10/10
1071/1071 [==============================] - 1s 951us/step - loss: 55619118523.2239
119/119 [==============================] - 0s 846us/step - loss: 54641139712.0000
Results: -55356425420.80 (1439258430.71) MSE
```

In [188…

```
estimator.fit(X_train, y_train)
prediction = estimator.predict(X_test)
```

```
Epoch 1/10
1190/1190 [==============================] - 1s 911us/step - loss: 55127846086.6096
Epoch 2/10
1190/1190 [==============================] - 1s 862us/step - loss: 54856059187.8018
Epoch 3/10
1190/1190 [==============================] - 1s 892us/step - loss: 54981760994.7674
Epoch 4/10
1190/1190 [==============================] - 1s 889us/step - loss: 54843262506.5592
Epoch 5/10
1190/1190 [==============================] - 1s 863us/step - loss: 55705735116.4131
Epoch 6/10
1190/1190 [==============================] - 1s 884us/step - loss: 55315307295.5970
Epoch 7/10
1190/1190 [==============================] - 1s 893us/step - loss: 55891650018.3375
Epoch 8/10
1190/1190 [==============================] - 1s 878us/step - loss: 55295449357.9715
Epoch 9/10
1190/1190 [==============================] - 1s 888us/step - loss: 55454267036.0504
Epoch 10/10
1190/1190 [==============================] - 1s 869us/step - loss: 56319818830.2401
510/510 [==============================] - 0s 689us/step
```

In [214…

```
mse = get_mse(y_test, prediction)
mad = get_mad(prediction)

print("MSE = %f" % (mse))
print("MAD = %f" % (mad))
```

```
MSE = 57811779353.847527
MAD = 220.994705
```