DAFTAR PUSTAKA

Amin. (2012). Sistem Deteksi Dini Hama Wereng Batang Coklat Menggunakan Jaringan Syaraf Tiruan Backpropagation (Skripsi) Universitas Negeri Semarang.

Assauri, & Sofyan. (2007). Manajemen Pemasaran, Rajawali Pers, Jakarta.

Chen, W. (2018). A Novel AdaBoost and CNN Base for Vehicle Classification. IEEE Access, 60445-60455.

Dreyfus, & Gerard. (2002). In Neural Network Methhodology and Application (pp. 3-4). OriginalFrencheditionpublishedbyEyrolles,Paris.

Fausett. (1994). Fundamentals of Neural Networks Architectures, Algorithms, and application.

Ghozali, & Imam. (2001). Aplikasi Analisis Multivariate dengan Program SPSS, Semarang: Badan Penerbit Universitas Diponegoro.

Ghozali, & Imam. (2005). Aplikasi Analisis Multivariat dengan program SPSS, Badan.

Hasan. (2003). Pokok-Pokok Materi Statistika 2 (Statistik Inferensia). Jakarta: Bumi Aksara.

Haykin, & Simon. (2009). Neural Networks and Learning Machines Third Edition. New Jersey: Pearson Education.

Indriaty, D. R. (2010). Analisis Pengaruh Tingkat Kualitas Pelayanan Jasa Puskesmas Terhadap Kepuasan Pasien (Skripsi), Universitas Diponegoro.

Khadija Jabeen, K. A. (2016). Abalone Age Prediction using Artificial Neural Network. IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 5, Ver. II (Sept - Oct. 2016), PP 34-38.

Kotler. (2008). Prinsip – prinsip pemasaran. Edisi Dua Belas. Jilid 1. Jakarta: Erlangga.

Kotler. (2009). Manajemen Pemasaran, PT.INDEKS Jakarta.

Montgomery, D. C., & Peck, E. A. (1992). Introduction to Linear Regression Analysis. New York: John Wiley & Sons, Inc.

ıda, A. (2010). Faktor – Faktor Yang Mempengaruhi Konsumen Dalam
embeli Rumah (Studi Kasus di Perumahan Bukit Semarang Baru)
niversitas Diponegoro. 35.

Sanda. (2003). Perkembangan Perumahan RealEstate,Kompas. 9.

Syafwan, H., Saputra, & Herman. (2016). Penerapan Jaringan Syaraf Tiruan Dalam Memprediksi Tingkat Pengangguran di Sumatra Utara Menggunakan Metode Backprapogation. Jurnal dan Teknologi Sistem Informasi Jurteksi Royal Vol.3, Nomor 1 ISSN 2407-1811, 39-40.

Tedjakusuma, & Muryani, R. d. (2001). Analisis Faktor-faktor yang Mempengaruhi Perilaku Konsumen Dalam Pembelian Air Minum Mineral Di Kotamadya Surabaya. Jurnal Penelitian Dinamika Sosial. Vol. 2. No. 3. Desember. Universitas Airlangga. Surabaya. ,. 48 – 58.

Umam, B. S. (2018). Data Mining dan Big Data Analytics Edisi 2 ISBN: 978-602-5414-76-3. Yogyakarta.

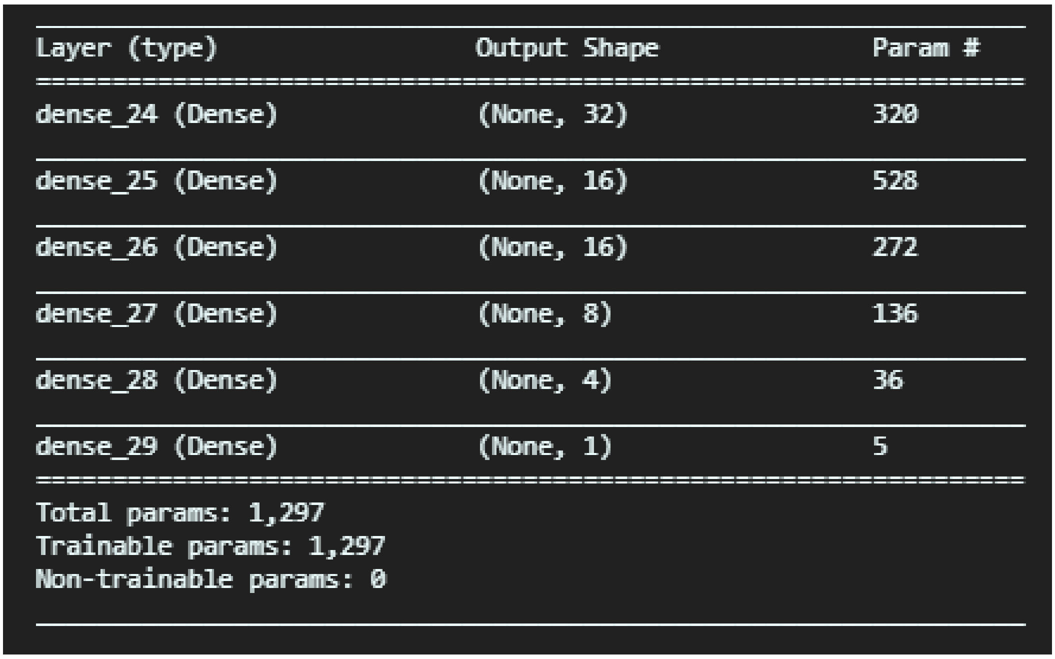
WIKIPEDIA. (2020, february 1). Kecerdasan Buatan. Retrieved from Jaringan saraf tiruan: https://id.wikipedia.org/wiki/Jaringan_saraf_tiruan

Witten H, I. E. (2001). Data Mining: Pratical Machine Learning Tools and Techniques, Third Edition.Burlington: Morgan Kaufmann Publishers.

Yunjiao Cai, Z. F. (2007). Comparison of Statistical Learning and Predictive Models on Breast Cancer Data and King County Housing Data.

Lampiran

#Jumlah parameter pada Artificial Neural Network.



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_24 (Dense)             (None, 32)                320

dense_25 (Dense)             (None, 16)                528

dense_26 (Dense)             (None, 16)                272

dense_27 (Dense)             (None, 8)                 136

dense_28 (Dense)             (None, 4)                 36

dense_29 (Dense)             (None, 1)                 5
=================================================================
Total params: 1,297
Trainable params: 1,297
Non-trainable params: 0
```

#parameter pada Regresi Linear Berganda.

Cross Validation 1

$Y = -33433 - 90598.6X_1 + 135031.7X_2 + 481580.4X_3 + 53613.29X_4 + 79232.77X_5 + 504650.3X_6 + 320390.2X_7 - 18894.2X_8 + 277963.6X_9$

Cross Validation 2

$Y = -335035 - 97540.1X_1 + 144818.8X_2 + 489454.7X_3 + 51743.05X_4 + 82199.52X_5 + 504496.7X_6 + 319590.6X_7 - 20882.6X_8 + 280029.7X_9$

Cross Validation 3

$Y = -334333 - 93628.5X_1 + 144192.4X_2 + 481580.4X_3 + 57320.19X_4 + 78432.45X_5 + 503873.8X_6 + 316483X_7 - 14915.6X_8 + 278802.9X_9$

Cross Validation 4

$Y = -339001 - 90897.9X_1 + 145224.2X_2 + 483661.9X_3 + 50909.02X_4 + 82761.6X_5 + 510008.3X_6 + 316798.9X_7 - 18269.2X_8 + 282238.4X_9$

lidation 5

8082-97010.9$X_1 + 145692.3X_2 + 484907.8X_3 + 52282.35X_4 +$

$1X_5 + 505611.1X_6 + 317669.4X_7 - 15004.5X_8 + 281963.1X_9$

#parameter pada Regresi Polinomial.

Cross Validation 1

$Y = -82219.6 - 217603X_1 - 57753.25X_2 + 160825.5X_3 + 8877.515X_4 +$
$60531.1X_5 - 254544X_6 + 848379.5X_7 + 310432.7X_8 -$
$290595X_9 + 95505.82X_1{}^2 - 139451X_1X_2 - 221386X_1X_3 + 118810.7X_1X_4 -$
$93422.4X_1X_5 + 201937.6X_1X_6 - 53082.6X_1X_7 + 234533X_1X_8 +$
$177898.4X_1X_9 - 27680.2X_2{}^2 + 414679.9X_2X_3 - 204855X_2X_4 -$
$136529X_2X_5 + 139790.5X_2X_6 + 71173.96X_2X_7 - 28776.2X_2X_8 -$
$42716.9X_2X_9 - 329487X_3{}^2 + 239250.2X_3X_4 + 281000.3X_3X_5 +$
$413902.1X_3X_6 + 336910.9X_3X_7 - 224270X_3X_8 - 75731.3X_3X_9 +$
$33181.58X_4{}^2 + 58910.3X_4X_5 - 50902.1X_4X_6 + 55989.35X_4X_7 -$
$184752X_4X_8 - 91144.6X_4X_9 - 32965.5X_5{}^2 + 84211.67X_5X_6 -$
$24699.7X_5X_7 + 60995.86X_5X_8 + 20980.29X_5X_9 + 183640.4X_6{}^2 +$
$363185.9X_6X_7 - 452668X_6X_8 + 337647X_6X_9 - 669596X_7{}^2 -$
$266880X_7X_8 - 320706X_7X_9 + 120737.2X_8{}^2 - 169863X_8X_9 + 146666.7X_9{}^2$

Cross Validation 2

$Y = -72587 - 193624X_1 - 65821.8X_2 + 185744.7X_3 + 73698.67X_4 +$
$101491.6X_5 - 305233X_6 + 861044X_7 + 317354.8X_8 - 288972X_9 +$
$70425.52X_1{}^2 - 108037X_1X_2 - 202692X_1X_3 + 135293.8X_1X_4 -$
$77842.5X_1X_5 + 173109.3X_1X_6 - 56500.5X_1X_7 + 166746.1X_1X_8 +$
$138627.1X_1X_9 + 38366.27X_2{}^2 + 298847.7X_2X_3 - 187368X_2X_4 -$
$169502X_2X_5 + 169502X_2X_6 + 55264.39X_2X_7 + 53849.16X_2X_8 +$
$45248.11X_2X_9 - 337512X_3{}^2 + 157377.7X_3X_4 + 307397.2X_3X_5 +$
$474628X_3X_6 + 342114.6X_3X_7 - 216442X_3X_8 - 118786X_3X_9 +$
$40116.1X_4{}^2 + 23116.18X_4X_5 - 108563X_4X_6 + 31121.31X_4X_7 -$
$181776X_4X_8 - 71057X_4X_9 - 57590.7X_5{}^2 + 66653.28X_5X_6 -$
$15699.2X_5X_7 + 64278.05X_5X_8 + 13468.32X_5X_9 + 188262.9X_6{}^2 +$
[illegible] $X_6X_7 - 471576X_6X_8 + 362582.2X_6X_9 - 697921X_7{}^2 -$
[illegible] $X_7X_8 + 288445.9X_7X_9 + 122946.5X_8{}^2 - 184299X_8X_9 + 154361X_9{}^2$

Cross Validation 3

$Y = -35437.5 - 205578X_1 - 88779.6X_2 + 21718.9X_3 - 5819.99X_4 +$
$96531.14X_5 - 273747X_6 + 827264.7X_7 + 272727.1X_8 -$
$364184X_9 + 70425.52{X_1}^2 - 15059.5X_1X_2 - 254952X_1X_3 + 128641.9X_1X_4 -$
$84939.2X_1X_5 + 147045.2X_1X_6 - 52848.1X_1X_7 + 209541X_1X_8 +$
$171252.3X_1X_9 + 17923.19{X_2}^2 + 300808.4X_2X_3 - 166295X_2X_4 -$
$155717X_2X_5 + 231940.7X_2X_6 + 65241.48X_2X_7 + 21861.85X_2X_8 +$
$35396.1X_2X_9 - 302282{X_3}^2 + 190845X_3X_4 + 282904.8X_3X_5 +$
$445041.4X_3X_6 + 342114.6X_3X_7 - 228812X_3X_8 - 138702X_3X_9 +$
$43178.4{X_4}^2 + 62511.47X_4X_5 - 83139.1X_4X_6 + 56947.78X_4X_7 -$
$144668X_4X_8 - 72181.2X_4X_9 - 49678.4{X_5}^2 + 31184.96X_5X_6 -$
$9026.18X_5X_7 + 80755.26X_5X_8 + 13975.62X_5X_9 + 188225.1{X_6}^2 +$
$350788.9X_6X_7 - 450173X_6X_8 + 385104.3X_6X_9 - 687393{X_7}^2 -$
$247520X_7X_8 + 319118.4X_7X_9 + 100648.5{X_8}^2 - 136082X_8X_9 + 174031.9{X_9}^2$

Cross Validation 4

$Y = -83151.6 - 201168X_1 - 42807.8X_2 + 102215X_3 + 21680.7X_4 +$
$152384.8X_5 - 267828X_6 + 894231X_7 + 343858.7X_8 -$
$332766X_9 + 70368.98{X_1}^2 + 11649.75X_1X_2 - 257457X_1X_3 + 114508.5X_1X_4 -$
$117290X_1X_5 + 19009.75X_1X_6 - 383481.1X_1X_7 + 159042.5X_1X_8 +$
$154719X_1X_9 - 30182.3{X_2}^2 + 422611.9X_2X_3 - 145927X_2X_4 -$
$205957X_2X_5 + 196365.3X_2X_6 + 13985.61X_2X_7 + 67872.47X_2X_8 +$
$34158.36X_2X_9 - 363841{X_3}^2 + 143967.7X_3X_4 + 349624X_3X_5 +$
$537373.4X_3X_6 + 332353.9X_3X_7 - 172457X_3X_8 - 69609.6X_3X_9 +$
$58347.61{X_4}^2 + 22042.13X_4X_5 - 103443X_4X_6 + 42980.22X_4X_7 -$
$153262X_4X_8 - 42046.7X_4X_9 - 62247.3{X_5}^2 + 83305.64X_5X_6 -$
$26007.2X_5X_7 + 19883.56X_5X_8 - 1275.81X_5X_9 + 171808.5{X_6}^2 +$
$371955.6X_6X_7 - 519774X_6X_8 + 305596.3X_6X_9 - 731098{X_7}^2 -$
$X_7X_8 + 325508.3X_7X_9 + 67188.8{X_8}^2 - 152178X_8X_9 + 166521.7{X_9}^2$

Cross Validation 5

$$\begin{aligned}
&\mathrm{Y} = -70427.8 - 211726X_1 -31607.9X_2 +93873.61X_3 + 1190857X_4 + \\
&95815.22X_5 - 242813X_6 + 870801.7X_7 +327520.2X_8 - \\
&315464X_9 +78631.87{X_1}^2 +5833.197X_1X_2 - 218817X_1X_3 + 83763.35X_1X_4 - \\
&94747.3X_1X_5 + 137987X_1X_6 - 58647.3X_1X_7 + 196621X_1X_8 + \\
&159373X_1X_9 - 46439.1{X_2}^2 + 379675X_2X_3 - 170347X_2X_4 - 103431X_2X_5 + \\
&238747.6X_2X_6 + 32977.67X_2X_7 + 3559.139X_2X_8 - 326661.1X_2X_9 - \\
&300335{X_3}^2 + 184813.8X_3X_4 + 289137.6X_3X_5 + 453848.3X_3X_6 + \\
&390812.1X_3X_7 - 224663X_3X_8 - 55232X_3X_9 + 53013.23{X_4}^2 + \\
&57943.68X_4X_5 - 100239X_4X_6 + 553137.59X_4X_7 - 129164X_4X_8 - \\
&58916.6X_4X_9 - 57348.6{X_5}^2 + 83125.32X_5X_6 - 34868.6X_5X_7 + \\
&37718.91X_5X_8 + 17120.08X_5X_9 + 165285.4{X_6}^2 + 338174.7X_6X_7 - \\
&453503X_6X_8 + 338216.8X_6X_9 - 702830{X_7}^2 - 274718X_7X_8 + \\
&321850.5X_7X_9 + 102393.2{X_8}^2 - 145439X_8X_9 + 155843.2{X_9}^2
\end{aligned}$$

```
#Library yang digunakan

import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model
from sklearn import metrics
from sklearn.feature_selection import RFE
from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import KFold
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.wrappers.scikit_learn import KerasRegressor
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import Adam
import statsmodels.api as sm
from scipy import stats

seed=10
tensorflow.random.set_seed(seed)

#membaca data
df = pd.read_csv('kc_house_data.csv', header=0)

#menampilkan data
df.head()

#menampilkan info data
df.info()

#menghapus variabel
df = df.drop(['id'], axis=1)

#menampilkan distribusi data
df1=df[['price', 'bedrooms', 'bathrooms', 'sqft_living', 'zipcode',
'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade', 'lat
', 'long',
    'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'sqft_l
iving15', 'sqft_lot15']]
h = df1.hist(bins=25,figsize=(16,16),xlabelsize='10',ylabelsize='10',x
rot=-15)
sns.despine(left=True, bottom=True)
[x.title.set_size(12) for x in h.ravel()];
[x.yaxis.tick_left() for x in h.ravel()];

#menambahkan variabel umur rumah dan umur renovasi rumah
df['sales_yr']=df['date'].astype(str).str[:4]
df['age']=df['sales_yr'].astype(int)-df['yr_built']

df['age_rnv']=0
df['age_rnv']=df['sales_yr'][df['yr_renovated']!=0].astype(int)-
df['yr_renovated'][df['yr_renovated']!=0]
df['age_rnv'][df['age_rnv'].isnull()]=0


bins = [-2,0,5,10,25,50,75,100,100000]
labels = ['<1','1-5','6-10','11-25','26-50','51-75','76-100','>100']
df['age_binned'] = pd.cut(df['age'], bins=bins, labels=labels)

bins = [-2,0,5,10,25,50,75,100000]
labels = ['<1','1-5','6-10','11-25','26-50','51-75','>75']
df['age_rnv_binned'] = pd.cut(df['age_rnv'], bins=bins, labels=labels)
df['sales_yr']=df['date'].astype(str).str[:4]

#menampilkan histogram umur rumah dan umur renovasi rumah
f, axes = plt.subplots(1, 2,figsize=(15,5))
p1=sns.countplot(df['age_binned'],ax=axes[0])
for p in p1.patches:
    height = p.get_height()
    p1.text(p.get_x()+p.get_width()/2,height + 50,height,ha="center")

p2=sns.countplot(df['age_rnv_binned'],ax=axes[1])
sns.despine(left=True, bottom=True)
for p in p2.patches:
    height = p.get_height()
    p2.text(p.get_x()+p.get_width()/2,height + 200,height,ha="center")

      ].set(xlabel='Age')
      ].yaxis.tick_left()
      ].yaxis.set_label_position("right")
```

```
axes[1].yaxis.tick_right()
axes[1].set(xlabel='Renovation Age')


#menampilkan variabel yang memiliki data nol.
print('zero values (in percent)')
for col in df:
    print(col,': ', (df[col]==0).sum()/len(df.index))

#menghapus variabel
df = df.drop(['waterfront','view','age_rnv'],axis=1)

#menampilkan uji f dan uji t
regressor_OLS = sm.OLS(df['price'], df.drop(['price'],axis=1)).fit()
regressor_OLS.summary()

#eliminasi variabel

y = df['price']
X = df.drop(['price'],axis=1)
colnames = X.columns
ranks = {}
def ranking(ranks, names, order=1):
    minmax = MinMaxScaler()
    ranks = minmax.fit_transform(order*np.array([ranks]).T).T[0]
    ranks = map(lambda x: round(x,2), ranks)
    return dict(zip(names, ranks))

lr = linear_model.LinearRegression(normalize=True,)
lr.fit(X,y)

rfe = RFE(lr, n_features_to_select=1, verbose=3)
rfe.fit(X,y)

lr = linear_model.LinearRegression(normalize=True)
lr.fit(X,y)


ridge = linear_model.Ridge(alpha = 7)
ridge.fit(X,y)

ranks['LR'] = ranking(np.abs(lr.coef_), colnames)
      'Ridge'] = ranking(np.abs(ridge.coef_), colnames)
      'RFE"] = ranking(list(map(float, rfe.ranking_)), colnames, order
#menampilkan hasil dari eliminasi variabel
r = {}
for name in colnames:
    r[name] = round(np.mean([ranks[method][name] for method in ranks.k
eys()]), 2)

meanplot = pd.DataFrame(list(r.items()), columns= ['Feature','Mean Ran
king'])

meanplot = meanplot.sort_values('Mean Ranking', ascending=False)

sns.factorplot(x="Mean Ranking", y="Feature", data = meanplot, kind="b
ar",
               size=14, aspect=1.9, palette='coolwarm')

#menghapus variabel dari hasil eliminasi variabel
df = df.drop(['sqft_lot', 'sqft_lot15', 'sqft_basement', 'sqft_living1
5', 'sqft_above','zipcode'], axis=1)



#minmaxscaler
minmax = MinMaxScaler(feature_range=(0.0, 1))
colnames = X.columns
X = minmax.fit_transform(X)
X = pd.DataFrame(X,columns=colnames)

#menampilkan hasil minmaxscaler
X.head()

#membuat model

#membuat model untuk menampilkan nilai r2 dan error pada ANN
def coeff_determination(y_true, y_pred):
    SS_res =  K.sum(K.square( y_true-y_pred ))
    SS_tot = K.sum(K.square( y_true - K.mean(y_true)))
    return ( 1 - SS_res/(SS_tot + K.epsilon()))

#membuat model hidden layer pada ANN
def baseline_model():
    model = Sequential()
    model.add(Dense(32, input_shape = (len(X.columns),), activation='r
elu'))
      del.add(Dense(16, activation='relu'))
      del.add(Dense(16, activation='relu'))
      del.add(Dense(8, activation='relu'))
      del.add(Dense(4, activation='relu'))
      del.add(Dense(1, activation='linear'))
    model.compile(loss='mse', optimizer=Adam(learning_rate=0.01), metr
ics=[coeff_determination])
    return model

#menampilkan jumlah parameter ANN
mlp.summary()

#menambahkan jumlah epoch dan batch size
EPOCHS = 500
BATCH_SIZE = 64

#menentukan Cross Validation sebanyak 5 kali.
kf = KFold(n_splits=5, shuffle=True, random_state=seed)
r2_cv = []
mse_cv = []

r2_cv_train = []
mse_cv_train = []

mlp_history = []
i=1

#iterasi cross validation
for train_index, test_index in kf.split(X):
    print('Cross Validation {}'.format(i))
    r2 = []
    mse = []
    r2_train = []
    mse_train = []

    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y[train_index], y[test_index]


    #membuat model regresi linear berganda
    lr = linear_model.LinearRegression().fit(X_train, y_train)
    #training score
    lr_y_pred = lr.predict(X_train)
    r2_train += [float(format(metrics.r2_score(y_train,lr_y_pred),'.
3f'))]
    mse_train += [float(format((metrics.mean_squared_error(y_train,lr_
y_pred)),'.3f'))]
         esting score
         _y_pred = lr.predict(X_test)
          += [float(format(metrics.r2_score(y_test,lr_y_pred),'.3f'))]
         e += [float(format((metrics.mean_squared_error(y_test,lr_y_pred)
         '))]
```

```
   # Membuat model regresi polinomial derajat 2
    poly2 = PolynomialFeatures(degree=2)
    Xpoly2 = poly2.fit_transform(X_train)
    poly2_lr = linear_model.LinearRegression().fit(Xpoly2, y_train)
    #training score
    poly2_y_pred = poly2_lr.predict(poly2.transform(X_train))
    r2_train += [float(format(metrics.r2_score(y_train,poly2_y_pred),'
.3f'))]
    mse_train += [float(format((metrics.mean_squared_error(y_train,pol
y2_y_pred)),'.3f'))]
    #testing score
    poly2_y_pred = poly2_lr.predict(poly2.transform(X_test))
    r2 += [float(format(metrics.r2_score(y_test,poly2_y_pred),'.3f'))]
    mse += [float(format((metrics.mean_squared_error(y_test,poly2_y_pr
ed)),'.3f'))]

    #Membuat model Articial Neural Network
    mlp = baseline_model()
    mlp_history += [mlp.fit(X_train,y_train, epochs=EPOCHS, batch_size
=BATCH_SIZE, verbose=2)]
    #mlp_history += [mlp.fit(X_train,y_train, epochs=EPOCHS, batch_siz
e=BATCH_SIZE, verbose=2).history['coeff_determination']]
    #training score
    mlp_y_pred = mlp.predict(X_train)
    r2_train += [float(format(metrics.r2_score(y_train,mlp_y_pred),'.3
f'))]
    mse_train += [float(format((metrics.mean_squared_error(y_train,mlp
_y_pred)),'.3f'))]
    #testing score
    mlp_y_pred = mlp.predict(X_test)
    r2 += [float(format(metrics.r2_score(y_test,mlp_y_pred),'.3f'))]
    mse += [float(format((metrics.mean_squared_error(y_test,mlp_y_pred
)),'.3f'))]

    r2_cv += [r2]
    mse_cv += [mse]
r2_cv_train += [r2_train]
    mse_cv_train += [mse_train]
    i +=1
mlp_history_loss = []
mlp_history_r2 = []
for i in range(len(mlp_history)):
      p_history_loss += [mlp_history[i].history['loss']]
      p_history_r2 += [mlp_history[i].history['coeff_determination']]
```

```
#menampilkan parameter rlb
pd.DataFrame(lr_coef_cv).T

#menampilkan parameter rp
pd.DataFrame(poly2_coef_cv).T

#Menampilkan jumlah parameter ANN
mlp = baseline_model()
mlp.summary()

#menyimpan hasil training yang digunakan untuk memprediksi data baru
lr_filename = "lr.save"
joblib.dump(lr, lr_filename)
poly2_filename = "poly2.save"
joblib.dump(poly2_lr, poly2_filename)
mlp_filename = "mlp.save"
mlp.save_weights(mlp_filename)

#menampilkan plot nilai error dari ANN
f, ax = plt.subplots()

ax.fill_between(range(1, EPOCHS+1), np.max(mlp_history_loss, axis=0),
np.min(mlp_history_loss,axis=0) , alpha=0.25, color="#00aaff")
ax.plot(range(1, EPOCHS+1), np.average(mlp_history_loss, axis=0), colo
r="#00aaff", label="Training Loss")
plt.ylabel('R-Squared')
plt.xlabel('epoch')
ax.legend()
ax.grid(True)

#Menampilkan plot nilai R2 dari ANN
f, ax = plt.subplots()

ax.fill_between(range(1, EPOCHS+1), np.max(mlp_history_r2, axis=0), np
.min(mlp_history_r2,axis=0) , alpha=0.25, color="#00aaff")

ax.plot(range(1, EPOCHS+1), np.average(mlp_history_r2, axis=0), color=
"#00aaff", label="Training R-Squared")
plt.ylabel('R-Squared')
plt.xlabel('epoch')
ax.legend()
ax.grid(True)

      ilkan hasil 5 kali cross validation nilai R2
      ing result
      sors = ['lr', 'poly2', 'mlp']
      nes = [1,2,3,4,5]
```

```
r2_cv_train = pd.DataFrame(r2_cv_train, columns=regressors)
r2_cv_train.index = row_names
r2_cv_train.index.name = 'cv'
r2_avg_train = np.average(r2_cv_train, axis=0)
r2_cv_train

# testing result
regressors = ['lr', 'poly2', 'mlp']
row_names = [1,2,3,4,5]
r2_cv = pd.DataFrame(r2_cv, columns=regressors)
r2_cv.index = row_names
r2_cv.index.name = 'cv'
r2_avg = np.average(r2_cv, axis=0)
r2_cv
#menampilkan hasil 5 kali cross validation nilai error

#training result
row_names = [1,2,3,4,5]
mse_cv_train = pd.DataFrame(mse_cv_train, columns=regressors)
mse_cv_train.index = row_names
mse_cv_train.index.name = 'cv'
mse_avg_train = np.average(mse_cv_train, axis=0)
mse_cv_train
#menampilkan RMSE
np.sqrt(mse_cv_train)
#testing result
row_names = [1,2,3,4,5]
mse_cv = pd.DataFrame(mse_cv, columns=regressors)
mse_cv.index = row_names
mse_cv.index.name = 'cv'
mse_avg = np.average(mse_cv, axis=0)
mse_cv
#Menampilkan RMSE
np.sqrt(mse_cv)

#merata-ratakan hasil 5 kali cross validation dari nilai R2 dan nilai
Error
#training
result_train = pd.DataFrame([r2_avg_train]+[np.sqrt(mse_avg_train)]).T
result_train.columns=['R2', 'RMSE']
result_train.index = regressors
result_train.index.name = 'regressor'
      _train
      ng
       = pd.DataFrame([r2_avg]+[np.sqrt(mse_avg)]).T
      .columns=['R2', 'RMSE']
```

```
result.index = regressors
result.index.name = 'regressor'
result
#memprediksi harga rumah dengan data baru

mlp_filename = "mlp.save"
poly2_filename = "poly2.save"
lr_filename = "lr.save"

mlp = baseline_model()
mlp.load_weights(mlp_filename)

lr = joblib.load(lr_filename)
poly2 = PolynomialFeatures(degree=2)
poly2_lr = joblib.load(poly2_filename)


#menampilkan data baru
data_baru = pd.read_excel('data_baru.xlsx')
colnames = data_baru.columns
data_baru

#minmaxscaler data baru
scaler_filename = "scaler.save"
minmax = MinMaxScaler()
minmax = joblib.load(scaler_filename)
data_baru = minmax.transform(data_baru)

#memprediksi harga jual rumah pada data baru
lr_pred = lr.predict(data_baru)
lr_pred

poly2_pred = poly2_lr.predict(poly2.fit_transform(data_baru))
poly2_pred


mlp_pred = mlp.predict(data_baru)
mlp_pred
```