Machine Learning Project - House Prices Prediction

Kai Liao

October 2020

1 Background

1.1 Introduction

House prices prediction is important for both buyers and sellers. As a economic student, I used to focus on estimating the hedonic price function. Hedonic regression breaks down the item being researched into its constituent characteristics, and obtains estimates of the contributory value of each characteristic. of houses. This method is derived from general equilibrium models and suitable for casual inference and counterfactual. However, this method is not designed for prediction of house prices. In this project, I will use the data of house prices in San Francisco, which was used by Bajari and Khan (2005) for hedonic regression, to train and evaluate machine learning models for houses price prediction.

1.2 Data Description

This project use San Francisco houses price dataset. This dataset contains 378252 observations. The list of features and summary statistic of the features are described in table 1 and table 2. The distribution of house prices distribution is given in figure 2.

To make the dataset suitable for linear regression and other machine learning algorithms. I made the 3 major changes. Firstly, I add square term of property crime rate, violent crime rate, year built, square footage, and number of total room. Secondly, I break down the features year of sell and county into dummies. To avoid multicollinearity, a value for each feature is not converted into a dummy. Thirdly, for neural network regression, I normalized the dataset to improve the performance of this algorithm. For linear regression, random forest, and decision tree, there is no need for normalization.

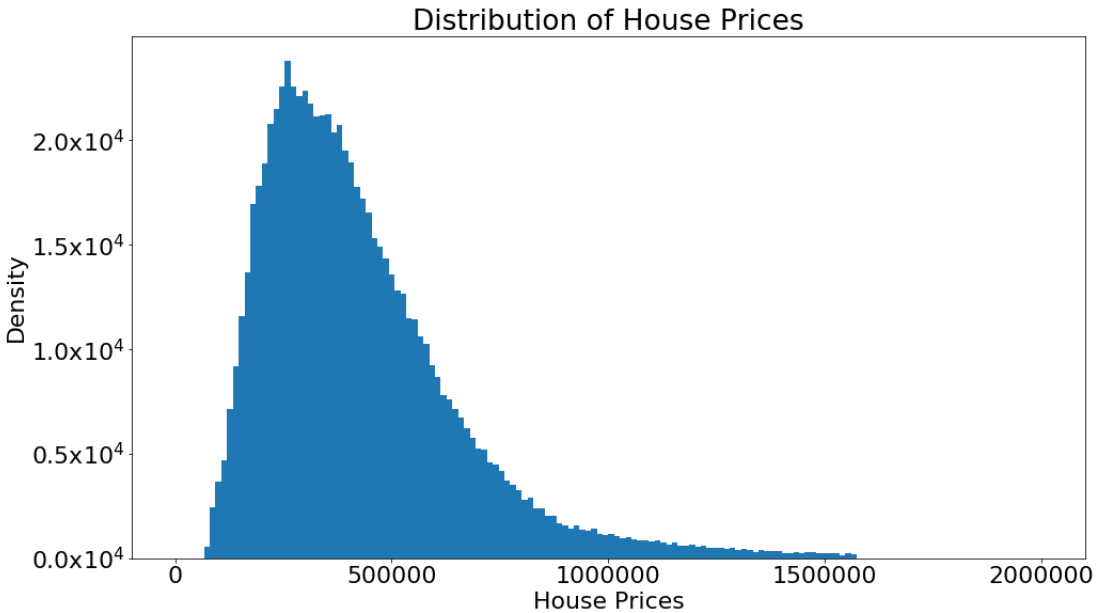
Table 1: Descriptive Statistics

	County	Price	Square Footage	Bathrooms	Bedrooms
Mean	All Samples	426931.9168	1701.671	2.129251	3.173004
Variance		53384150033	470894.8	0.52026	0.886927
Mean	Alameda	398873.1171	1672.362	2.031492	3.172366
Variance		41544980702	437987.6	0.572418	0.772822
Mean	Contra Costa	374582.3608	1833.148	2.247798	3.303357
Variance		46561031579	563350.1	0.500374	0.83612
Mean	San Francisco	501817.0863	1497.933	1.731226	2.618241
Variance		65620319073	365112.2	0.657902	1.021395
Mean	San Mateo	503715.0083	1587.405	2.04415	2.739668
Variance		73286582870	466259.4	0.565177	0.977258
Mean	Santa Clara	464683.1897	1662.787	2.161859	3.214504
Variance		55932151423	410670.7	0.440547	0.892107

Table 2: Descriptive Statistics Con't

	County	Total Rooms	Stories	Violent Crime Rate	Property Crime Rate
Mean	All Samples	6.987069	1.413098	389.9648	1718.545
Variance		3.945554	0.261747	46609.33	413696.2
Mean	Alameda	6.506236	1.434175	441.4993	1971.128
Variance		2.469336	0.260794	39922.4	371046.5
Mean	Contra Costa	8.014587	1.432249	415.7833	1898.75
Variance		4.266372	0.245412	77245.92	359813.9
Mean	San Francisco	5.867504	1.341677	586.1629	2141.654
Variance		3.493864	0.312684	43581.41	122186.4
Mean	San Mateo	5.934	1.355573	349.0791	1713.204
Variance		3.636281	0.25109	42814.56	1468430
Mean	Santa Clara	6.847879	1.401985	322.5581	1358.683
Variance		3.48604	0.270534	17379.23	80485.57

Figure 1:



1.3 Previous Work

This dataset is also used by Bajari and Khan (2005) in their research for estimating housing demand and explaining racial segregation in cities. They use linear regression to breaks down housing price into its attributes, e.g., number of bedrooms, footage, etc..This regression result is the first stage of their empirical specification and were further used to recover structural parameters of economic models. This method was first introduced by Rosen (1974). I will further replicate their work and discuss the theoretical base of this method in the next chapter.

Bajari and Khan (2005) also used non-parametric method to recover the hedonic price function for housing price. However, this method can only study the marginal value of one attribute each time. For example, if we want to estimate the effect of marginal increase in violent crime rate VC on house prices P. By choosing the grid of violent crime rate χ, we can solve for parameters $\alpha(\chi)$ and $\beta(\chi)$ by

$$\min_{\{\alpha(\chi),\beta(\chi)\}} \sum_{j=1}^{J} \left(P_j - \alpha(\chi) - \beta(\chi)VC_j\right)^2 K_h\left(VC_j - \chi\right) \tag{1}$$

where kernel K is given by

$$K_h\left(VC_j - \chi\right) = \frac{1}{h\sigma_{VC}} \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{VC_j - \chi}{h\sigma_{VC}}\right)^2\right\} \tag{2}$$

In my project, I will use other machine learning methods, including neural network, decision trees, and random forest. Although these methods can be hardly used to causal inference and counterfactual, they are good for making predictions. I will discuss the detail advantage of these methods in next chapter.

2 Methods

2.1 Linear Regression

The linear regression model is given by:

$$\begin{aligned} Price =& \beta_0 Constant + \beta_1 Bathrooms + \beta_2 Bedrooms + \beta_3 Stories + \beta_4 PropertyCrimeRate + \\ & \beta_5 (PropertyCrimeRate)^2 + \beta_6 YearBuilt + \beta_7 (YearBuilt)^2 + \beta_8 SquareFootage + \\ & \beta_9 (SquareFootage)^2 + \beta_{10} TotalRooms + \beta_{11} (TotalRooms)^2 + \beta_{12} ViolentCrimeRate \\ & + \beta_{13} (ViolentCrimeRate)^2 + \gamma_1 VectorofYearDummies + \gamma_2 VectorofCountyDummies \end{aligned} \tag{3}$$

This method have several unique advantages:

- Linear regression is interpretable. This model is derived from consumers and suppliers theories. It has a concrete theoretical base for interpretation. For example, if we take the first order partial derivative of violent crime rate (VC). We will have the price gradient for violent rate.

$$\frac{\partial Price}{\partial VC} = \beta_{12} + \beta_{13} VC \tag{4}$$

  Results like this can be used to second stage analysis and counterfactual.

- Linear regression bases on testable assumptions.

- This model has high computational efficiency and very easy to implement.

The most obvious disadvantage of this method is

- The prediction performance is bad compared to other machine learning algorithms. I use linear regression as one of the baseline to evaluate other algorithms. Other algorithms should be at least as good as linear regression.

2.2 Neural Network

Neural network is a series of algorithms that are constructed as systems of neurons to recognize underlying relationships in a set of data.

The advantages are

- It has better ability to make prediction when the parameters are correctly tuned.
- Keras, a python deep learning API, provide a good solution to neural network problem. It also have high computational efficiency.

2.3 Decision Tree

The decision tree algorithm constructs a tree to classify data. It predict the outcome by looking at features from the top to the bottom of the tree. The advantages are

- It is interpretable and intuitive.
- Sklearn, a python deep learning API, provide a good solution to decision tree problem. It also have high computational efficiency.

2.4 Random Forest

The random forest algorithm constructs trees to classify data. The advantages are

- Random forest normally has better accuracy than decision tree algorithm.
- Sklearn provide a good solution to random forest problem. It also have high computational efficiency.

3 Experiments

3.1 Models and Hyperparameter Selection

3.1.1 Neural Network

I use a sequential neural network model. I add 5 layers to this model and let the number of neurons of each model be 33, because the data have 33 features. These layers are equipped with relu activation functions. The final layer will predict a continuous value. I choose mean squared error as the loss function, which is identical with linear regression.

```
def creat_model(layer = 5):
    clt_nn = Sequential()
    for i in range(layer):
        clt_nn.add(Dense(33,activation='relu'))
    clt_nn.add(Dense(1))
    clt_nn.compile(optimizer='Adam',loss='mse')
    return clt_nn
```

I tune the hyperparameter *epochs* by 5-fold cross validation. The package Keras provide a built-in way to return error on testing set for each epoch. I take the means for 5-fold cross validation for each epoch and find the epoch that gives the minimum of the error on testing set. The best epoch is 12.

```
EPOCH_MAX = 100
epochs = list(range(1,EPOCH_MAX+1))
val_loss = []
val_loss_mean = []
```

```
for train_index, test_index in kf.split(X_train):
    X_train_cross, X_test_cross = X_train_df.loc[train_index], X_train_df.loc[test_index]
    y_train_cross, y_test_cross = y_train_df.loc[train_index], y_train_df.loc[test_index]
    clt_nn = creat_model(5)
    history = clt_nn.fit(x=X_train_cross,y=y_train_cross,
          validation_data=(X_test_cross,y_test_cross),
          batch_size=256,epochs=EPOCH_MAX)
    val_loss.append(history.history['val_loss'])
val_loss_mean.append(np.mean(val_loss, axis=0))
# find the best epoch
best_epoch = np.argmin(val_loss_mean[0])+1
```

Lastly, I train the model with the entire training set with the tuned hyperparameter and then make prediction of the testing set with the trained model.

```
# Train the model again on the entire testing set with the best parameters.
clt_nn = creat_model(5)
clt_nn.fit(x=X_train,y=y_train, validation_data=(X_test,y_test),
            batch_size=256,epochs=best_epoch)
# Make Prediction on Testing Set
y_pred_nn = clt_nn.predict(X_test)
```

3.1.2 Decision Tree

The hyperparameter I tune for the decision tree model is *max_depth*. I use GridSearchCV of Sklearn and 5-fold cross validation for tuning. For computation efficiency, I only grid search the value of [5,10,15,20,25]. The best *max_depth* is 10. I then use this parameter to train the decision tree and make prediction.

```
# define the tuning space
params_grid = [{'max_depth':[5,10,15,20,25]}]
# tuning max_depth
clf_dt_grid = GridSearchCV(DecisionTreeRegressor(), params_grid, cv=5)
clf_dt_grid.fit(X_train, y_train)
# traing with the best parameter
clf_dt = clf_dt_grid.best_estimator_
# prediction
y_pred_dt = clf_dt.predict(X_test)
```

3.1.3 Random Forest

I also tune the *max_depth* for random forest. The best *max_depth* is 25. The procedure of tuning, training, and making prediction is similar to the decision tree.

```
# define the tuning space
params_grid = [{'max_depth':[5,10,15,20,25]}]
# tuning max_depth
clf_rf_grid = GridSearchCV(RandomForestRegressor(), params_grid, cv=5)
clf_rf_grid.fit(X_train, y_train)
# traing with the best parameter
clf_rf = clf_rf_grid.best_estimator_
# prediction
y_pred_rf = clf_rf.predict(X_test)
```

4 Results and Evaluations

I calculate the normalized prediction error NE of house i by

$$NE_i = \frac{|Predicted\_Houses\_Price_i - Houses\_Price_i|}{Houses\_Price_i} \tag{5}$$

Then, I plot the histgrams of normalized prediction error of each algorithm.

Figure 2:

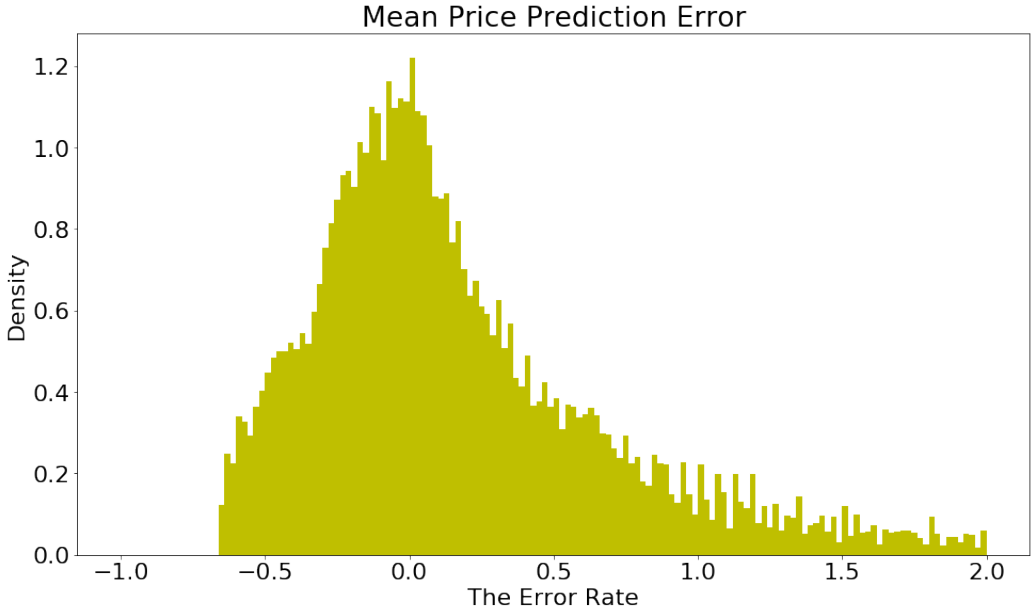


Figure 3:

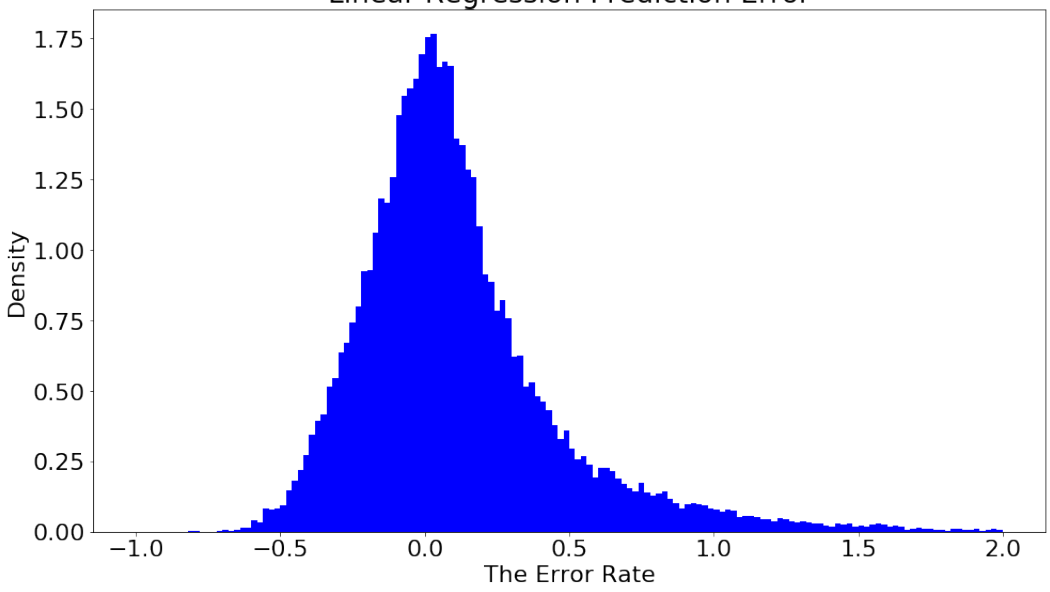


Figure 4:

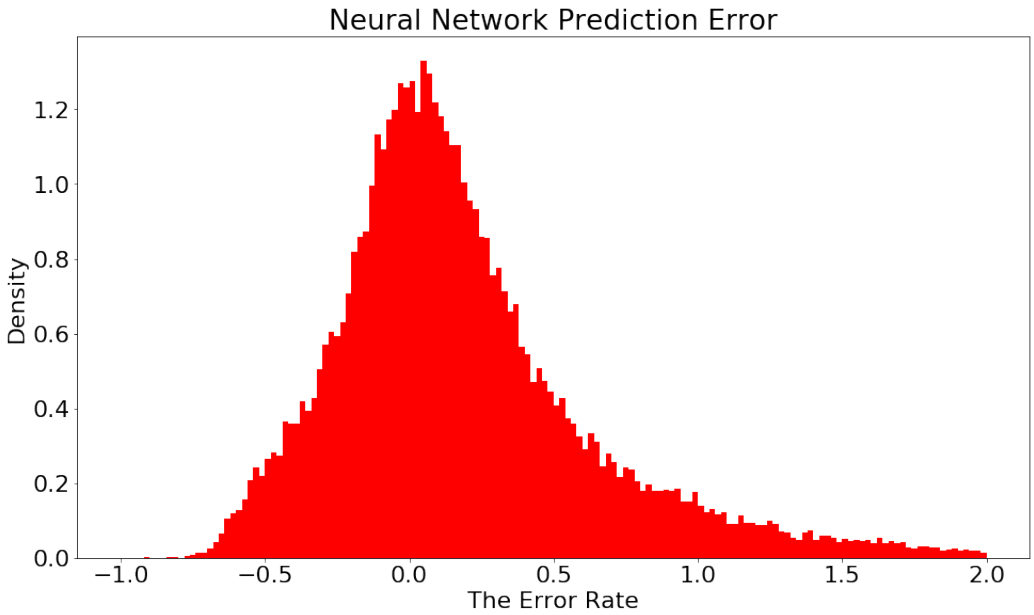


Figure 5:

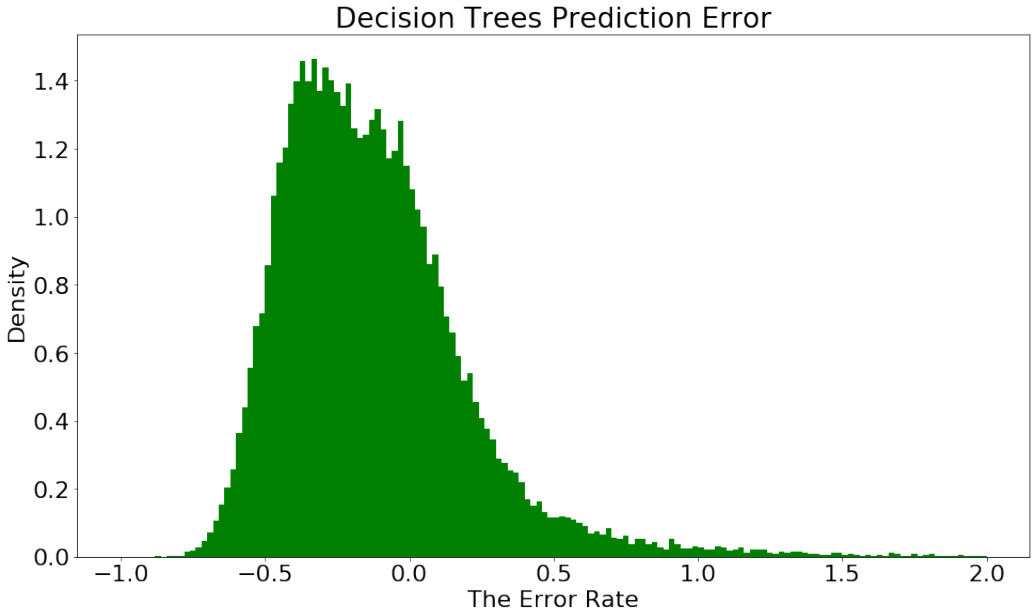


Figure 6:

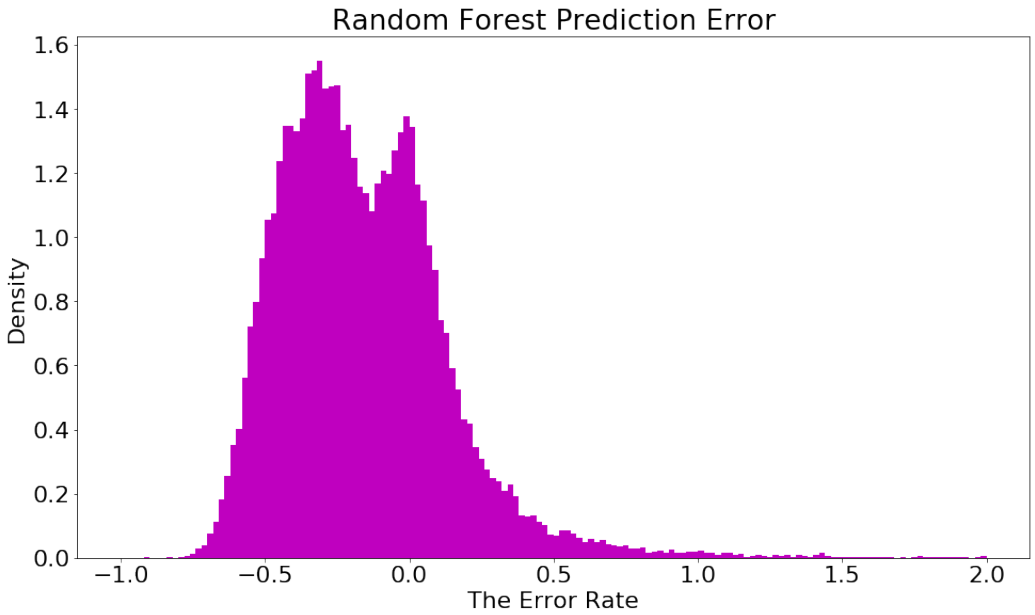
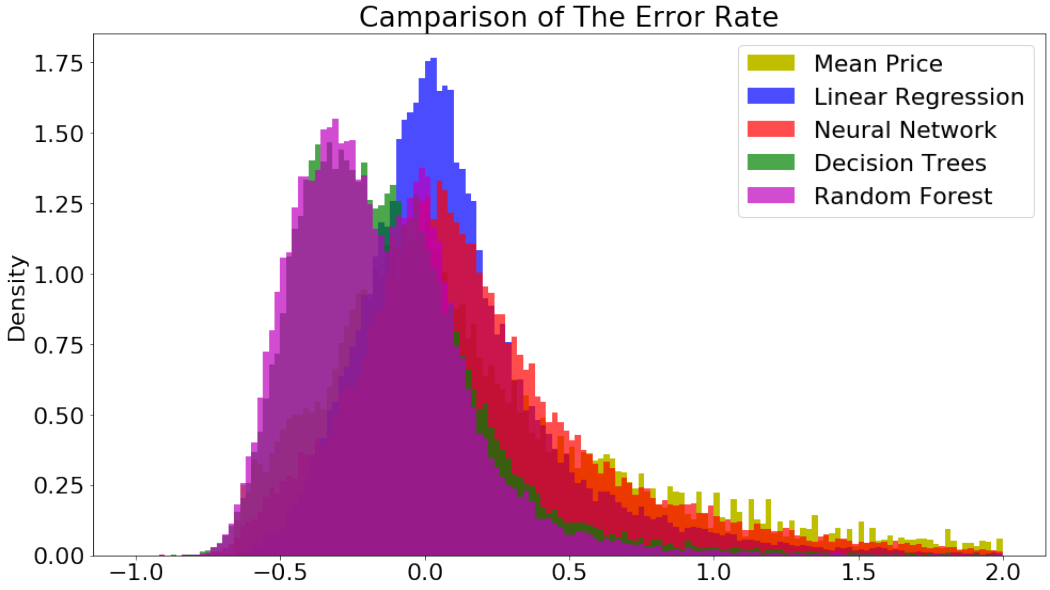


Figure 7:

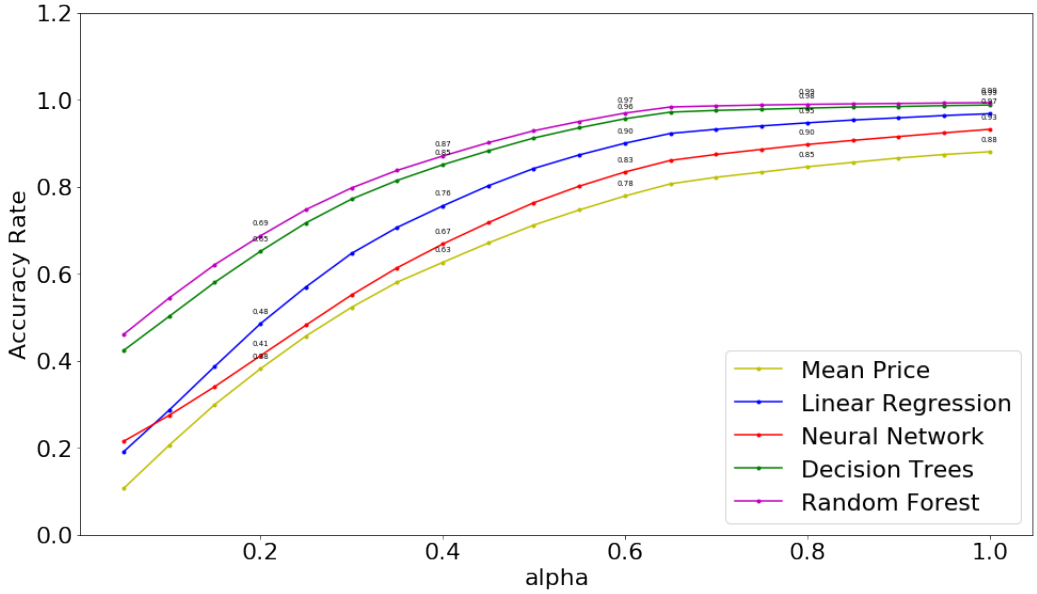


Given a threshold α, I calculate the accuracy rate AR by

$$AR = \frac{\sum_i^N \mathbb{1}[-\alpha < NE_i \leq \alpha]}{N} \tag{6}$$

Figure 7 shows the result of accuracy rate of each algorithm. Generally speaking, the prediction accuracy can be rank by: **Mean Price** < **Linear Regression** < **Neural Network** < **Decision Trees** < **Random Forest**. Set $\alpha = 0, 2$, the accuracy rate is 0.38, 0.41, 0.48, 0.65, 0.69 correspondingly. If we choose linear regression as the baseline model, the prediction accuracy of the best alternative, random forest, increase 0.27, or 65.8%. This shows a great improvement of prediction ability compared to previous economics papers.

Figure 8:



5 Reviews

In this chapter, I am going to briefly talk about what can be improved in this model.

- ***Tuning More Hyperparameters.*** Due to the computation constrain, I do not tune more than one parameter for each algorithm. However, there are many hyperparameters to be tuned. Especially for the neural network model, other than number of epochs, batch size, training optimization algorithm, learning rate, neuron activation function, number of neurons in the hidden layer, and number of layers also need to be tuned. In my implementation, I choose batch size, neuron activation function, and number of neurons base on the characteristic of my dataset, and only tune number of epochs.

- ***Dataset with More Features.*** My dataset has relatively small feature space but includes a large number of samples. It is more suitable for estimating complicated structural model, but less effective to make prediction. If I have more information on other features, the accuracy will increase.

Reference:

Bajari, P., & Kahn, M. (2005). Estimating Housing Demand with an Application to Explaining Racial Segregation in Cities. *Journal of Business Economic Statistics*, 23(1), 20-33.

Timmins, S. (2020). Housing Transactions for the Los Angeles and San Francisco Metropolitan Areas Between 1993 and 2008

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statistics import mean

pd.set_option('display.max_columns', None)
```

In [2]:
```
# Import Data
data_df = pd.read_csv('sf_data.csv', header = None)
```

1 Data Cleaning

1.1 Feature Space

Each data set contains the following Features (in order):

- House ID #
- Price (deflated to year 2000 dollars)
- county ID # (see below)
- Year Built
- Square Footage
- # Bathrooms
- # Bedrooms
- # Total Rooms
- # Stories
- Violent Crime Rate (Cases per 100,000)
- Property Crime Rate (Cases Per 100,000)
- Year of Sale (1993-2008)

In [3]:
```
# Rename the columns
data_df.rename(columns={0: 'House_ID', 1: 'Price', 2: 'County_ID',
                        3: 'Year_Built', 4: 'Square_Footage', 5: 'Bathrooms',
                        6: 'Bedrooms', 7: 'Rooms', 8: 'Stories',
                        9: 'VC', 10: 'PC', 11: 'Year_of_Sale'}, inplace=True)
```

1.2 Construct Dummies for Counties

- 1 Alameda
- 13 Contra costa
- 75 San Francisco
- 81 San Mateo
- 85 Santa Clara

For San Francisco, I include dummies for counties 13, 75, 81, and 85. Alameda is the baseline.

In [4]:
```
for label, county in enumerate(set(data_df.iloc[:]['County_ID'])):
    for i in range(len(data_df.index)):
        if data_df.iloc[i]['County_ID'] == county:
            data_df.at[i,str(county)] = 1
```

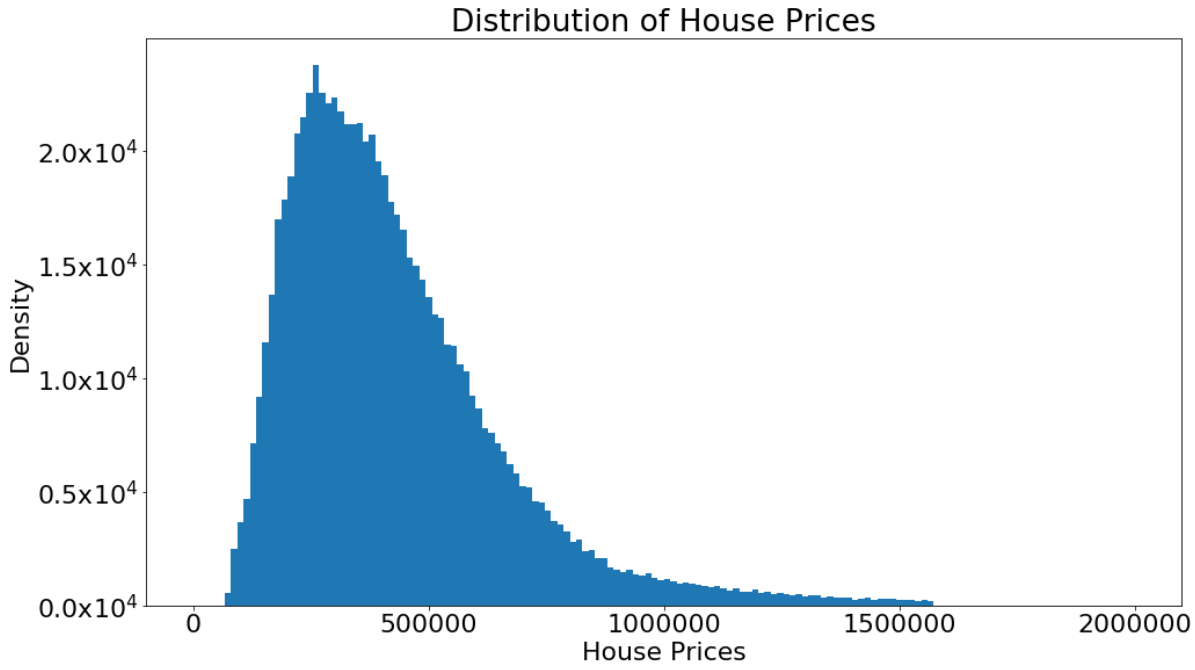
In [5]:
```
data_df = data_df.fillna(0)
```

In [6]:
```
data_df.to_csv('data_modified.csv')
```

1.3 Discriptive Statistic

In [7]:
```
from scipy.stats import norm
from sklearn.neighbors import KernelDensity
from matplotlib.pyplot import MultipleLocator
from pylab import rcParams
from matplotlib.ticker import FuncFormatter
rcParams['figure.figsize'] = 16, 9
plt.rcParams.update({'font.size': 22})
```

In [12]:
```
plt.hist(y, bins=150, range = (1,2000000), density = True,
         label = 'Price')
plt.ylabel('Density')
plt.xlabel('House Prices')
plt.title('Distribution of House Prices')

x_major_locator=MultipleLocator(500000)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
def formatnum(x, pos):
    return '$%.1f$x$10^{4}$' % (x*1000000)
def formatnum_x(x, pos):
    return '$%.2f \pi$' % (x)
formatter1 = FuncFormatter(formatnum)
formatter2 = FuncFormatter(formatnum_x)
ax.yaxis.set_major_formatter(formatter1)
plt.savefig('Distribution of House Prices.png')
plt.show()
```



2 Models

The model assumption and economic theoretical base for the model is given in the write-up.

2.1 Linear Regression

2.1.1 Construct Variables for Regression

The linear regression model

- Constant
- # Bathrooms
- # Bedrooms
- # Stories
- Property Crime Rate
- (Property Crime Rate) ^{2}
- Year Built
- (Year Built) ^{2}
- Square Footage
- (Square Footage) ^{2}
- # Total Rooms
- (# Total Rooms) ^{2}
- Violent Crime Rate
- (Violent Crime Rate) ^{2}
- Vector of Year Dummies (omit 1999)
- Vector of Dummies for Certain Counties ^{2}

2.1.2 Conduct Linear Regression on Training Set

The first two parts are written in my Fortran code. They following python code aim to read the regression result and make prediction on testing set.

2.1.3 Make Prediction on Testing Set

In [9]:
```
# Read dataset
X_df = pd.read_csv('X_SF.csv')
y_df = pd.read_csv('y_SF.csv', header = None)
y_df.rename(columns={0: 'Price'}, inplace=True)
```

In [10]:
```
# Read regression results
reg_result = pd.read_csv('Hedonic_Price_Function_Regression_Results_SF.csv')
```

In [11]:
```
X = X_df.to_numpy()
y = y_df.to_numpy()
beta = reg_result.to_numpy()[0][:]

# Construct Testing Set (10% of the all samples)
X_test = X[340426:][:]
y_test = y[340426:][:]
```

In [14]:
```
# Make predictions
y_pred = np.matmul(X_test, beta.T)
```

2.2 Keras Regressions

2.2.1 Seperate Dataset

In [64]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import preprocessing
from sklearn.model_selection import KFold
```

In [59]:
```
# Nomalized the features

X = preprocessing.scale(X)
```

In [55]:
```
X_test = X[340426:][:]
y_test = y[340426:]
X_train = X[:340426][:]
y_train = y[:340426]
```

2.2.2 Train the Neural Network Model with Cross Validation

I cross validate the number of epochs times.

In [ ]:
```
def creat_model(layer = 5):
    clt_nn = Sequential()
    for i in range(layer):
        clt_nn.add(Dense(33,activation='relu'))
    clt_nn.add(Dense(1))
    clt_nn.compile(optimizer='Adam',loss='mse')
    return clt_nn
```

In [104]:
```
X_train_df = pd.DataFrame(data=X_train)
y_train_df = pd.DataFrame(data=y_train)
```

In [106]:
```
EPOCH_MAX = 100
epochs = list(range(1,EPOCH_MAX+1))
val_loss = []
val_loss_mean = []

for train_index, test_index in kf.split(X_train):
    X_train_cross, X_test_cross = X_train_df.loc[train_index], X_train_df.loc[
test_index]
    y_train_cross, y_test_cross = y_train_df.loc[train_index], y_train_df.loc[
test_index]
    clt_nn = creat_model(5)
    history = clt_nn.fit(x=X_train_cross,y=y_train_cross,
          validation_data=(X_test_cross,y_test_cross),
          batch_size=256,epochs=EPOCH_MAX)
    val_loss.append(history.history['val_loss'])
val_loss_mean.append(np.mean(val_loss, axis=0))
```

```
Epoch 1/100
WARNING:tensorflow:Layer dense_195 is casting an input tensor from dtype floa
t64 to the layer's dtype of float32, which is new behavior in TensorFlow 2.
The layer has dtype float32 because its dtype defaults to floatx.

If you intended to run this layer in float32, you can safely ignore this warn
ing. If in doubt, this warning is likely only an issue if you are porting a T
ensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backen
d.set_floatx('float64')`. To change just this layer, pass dtype='float64' to
the layer constructor. If you are the author of this layer, you can disable a
utocasting by passing autocast=False to the base Layer constructor.

1064/1064 [==============================] - 1s 1ms/step - loss: 61263572992.
0000 - val_loss: 68190650368.0000
Epoch 2/100
1064/1064 [==============================] - 1s 934us/step - loss: 1634299596
8.0000 - val_loss: 64050769920.0000
Epoch 3/100
1064/1064 [==============================] - 1s 937us/step - loss: 1533058048
0.0000 - val_loss: 52238340096.0000
Epoch 4/100
1064/1064 [==============================] - 1s 933us/step - loss: 1465024102
4.0000 - val_loss: 36747956224.0000
Epoch 5/100
1064/1064 [==============================] - 1s 935us/step - loss: 1417506918
4.0000 - val_loss: 26531129344.0000
Epoch 6/100
1064/1064 [==============================] - 1s 912us/step - loss: 1385683660
8.0000 - val_loss: 19664203776.0000
Epoch 7/100
1064/1064 [==============================] - 1s 951us/step - loss: 1357751603
2.0000 - val_loss: 16738002944.0000
Epoch 8/100
1064/1064 [==============================] - 1s 935us/step - loss: 1334510796
8.0000 - val_loss: 15927662592.0000
Epoch 9/100
1064/1064 [==============================] - 1s 927us/step - loss: 1315805798
4.0000 - val_loss: 16780314624.0000
Epoch 10/100
1064/1064 [==============================] - 1s 959us/step - loss: 1298764902
4.0000 - val_loss: 18297030656.0000
Epoch 11/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12839964672.
0000 - val_loss: 20837089280.0000
Epoch 12/100
1064/1064 [==============================] - 1s 922us/step - loss: 1269977702
4.0000 - val_loss: 23985649664.0000
Epoch 13/100
1064/1064 [==============================] - 1s 929us/step - loss: 1257812582
4.0000 - val_loss: 30214993920.0000
Epoch 14/100
1064/1064 [==============================] - 1s 984us/step - loss: 1246960844
8.0000 - val_loss: 32907278336.0000
Epoch 15/100
1064/1064 [==============================] - 1s 985us/step - loss: 1237562163
```

```
2.0000 - val_loss: 38302056448.0000
Epoch 16/100
1064/1064 [==============================] - 1s 936us/step - loss: 1230851788
8.0000 - val_loss: 46019731456.0000
Epoch 17/100
1064/1064 [==============================] - 1s 958us/step - loss: 1222913024
0.0000 - val_loss: 50010103808.0000
Epoch 18/100
1064/1064 [==============================] - 1s 937us/step - loss: 1216188518
4.0000 - val_loss: 59369287680.0000
Epoch 19/100
1064/1064 [==============================] - 1s 957us/step - loss: 1210091417
6.0000 - val_loss: 61903675392.0000
Epoch 20/100
1064/1064 [==============================] - 1s 912us/step - loss: 1204882944
0.0000 - val_loss: 68002013184.0000
Epoch 21/100
1064/1064 [==============================] - 1s 915us/step - loss: 1199496704
0.0000 - val_loss: 73518505984.0000
Epoch 22/100
1064/1064 [==============================] - 1s 915us/step - loss: 1195647488
0.0000 - val_loss: 76439609344.0000
Epoch 23/100
1064/1064 [==============================] - 1s 921us/step - loss: 1190723481
6.0000 - val_loss: 81414242304.0000
Epoch 24/100
1064/1064 [==============================] - 1s 914us/step - loss: 1185820057
6.0000 - val_loss: 90551394304.0000
Epoch 25/100
1064/1064 [==============================] - 1s 931us/step - loss: 1182389145
6.0000 - val_loss: 95207858176.0000
Epoch 26/100
1064/1064 [==============================] - 1s 922us/step - loss: 1178048000
0.0000 - val_loss: 94252490752.0000
Epoch 27/100
1064/1064 [==============================] - 1s 916us/step - loss: 1175012044
8.0000 - val_loss: 100071931904.0000
Epoch 28/100
1064/1064 [==============================] - 1s 927us/step - loss: 1170868531
2.0000 - val_loss: 103588855808.0000
Epoch 29/100
1064/1064 [==============================] - 1s 911us/step - loss: 1167751680
0.0000 - val_loss: 114145607680.0000
Epoch 30/100
1064/1064 [==============================] - 1s 925us/step - loss: 1164465971
2.0000 - val_loss: 114028797952.0000
Epoch 31/100
1064/1064 [==============================] - 1s 926us/step - loss: 1162503577
6.0000 - val_loss: 116933025792.0000
Epoch 32/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11605429248.
0000 - val_loss: 121678831616.0000
Epoch 33/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11583286272.
0000 - val_loss: 122603520000.0000
Epoch 34/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11559271424.
```

```
0000 - val_loss: 121354805248.0000
Epoch 35/100
1064/1064 [==============================] - 2s 2ms/step - loss: 11537476608.
0000 - val_loss: 124140240896.0000
Epoch 36/100
1064/1064 [==============================] - 2s 1ms/step - loss: 11524714496.
0000 - val_loss: 130942484480.0000
Epoch 37/100
1064/1064 [==============================] - 2s 1ms/step - loss: 11499382784.
0000 - val_loss: 136130707456.0000
Epoch 38/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11479267328.
0000 - val_loss: 136366014464.0000
Epoch 39/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11464581120.
0000 - val_loss: 146776080384.0000
Epoch 40/100
1064/1064 [==============================] - 2s 2ms/step - loss: 11438174208.
0000 - val_loss: 141004587008.0000
Epoch 41/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11432287232.
0000 - val_loss: 137711648768.0000
Epoch 42/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11415011328.
0000 - val_loss: 143306686464.0000
Epoch 43/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11394386944.
0000 - val_loss: 135824949248.0000
Epoch 44/100
1064/1064 [==============================] - 2s 2ms/step - loss: 11375207424.
0000 - val_loss: 140317605888.0000
Epoch 45/100
1064/1064 [==============================] - 3s 2ms/step - loss: 11353631744.
0000 - val_loss: 142990786560.0000
Epoch 46/100
1064/1064 [==============================] - 2s 1ms/step - loss: 11337247744.
0000 - val_loss: 143451865088.0000
Epoch 47/100
1064/1064 [==============================] - 2s 2ms/step - loss: 11314747392.
0000 - val_loss: 151173169152.0000
Epoch 48/100
1064/1064 [==============================] - 2s 2ms/step - loss: 11299586048.
0000 - val_loss: 145731878912.0000
Epoch 49/100
1064/1064 [==============================] - 2s 2ms/step - loss: 11284261888.
0000 - val_loss: 157875273728.0000
Epoch 50/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11266202624.
0000 - val_loss: 153581731840.0000
Epoch 51/100
1064/1064 [==============================] - 2s 2ms/step - loss: 11259620352.
0000 - val_loss: 141104414720.0000
Epoch 52/100
1064/1064 [==============================] - 2s 1ms/step - loss: 11241334784.
0000 - val_loss: 142481817600.0000
Epoch 53/100
1064/1064 [==============================] - 2s 2ms/step - loss: 11229539328.
```

```
0000 - val_loss: 147683917824.0000
Epoch 54/100
1064/1064 [==============================] - 2s 1ms/step - loss: 11210594304.
0000 - val_loss: 138389061632.0000
Epoch 55/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11199271936.
0000 - val_loss: 152848826368.0000
Epoch 56/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11192562688.
0000 - val_loss: 143901605888.0000
Epoch 57/100
1064/1064 [==============================] - 1s 995us/step - loss: 1117840691
2.0000 - val_loss: 143765798912.0000
Epoch 58/100
1064/1064 [==============================] - 1s 933us/step - loss: 1117481369
6.0000 - val_loss: 153442205696.0000
Epoch 59/100
1064/1064 [==============================] - 1s 940us/step - loss: 1115451494
4.0000 - val_loss: 152867012608.0000
Epoch 60/100
1064/1064 [==============================] - 1s 909us/step - loss: 1115556147
2.0000 - val_loss: 150288384000.0000
Epoch 61/100
1064/1064 [==============================] - 1s 974us/step - loss: 1113211801
6.0000 - val_loss: 157044146176.0000
Epoch 62/100
1064/1064 [==============================] - 1s 957us/step - loss: 1112732057
6.0000 - val_loss: 157934616576.0000
Epoch 63/100
1064/1064 [==============================] - 1s 945us/step - loss: 1110690611
2.0000 - val_loss: 161147076608.0000
Epoch 64/100
1064/1064 [==============================] - 1s 982us/step - loss: 1110895001
6.0000 - val_loss: 163586179072.0000
Epoch 65/100
1064/1064 [==============================] - 1s 916us/step - loss: 1108645171
2.0000 - val_loss: 167602110464.0000
Epoch 66/100
1064/1064 [==============================] - 1s 915us/step - loss: 1108934860
8.0000 - val_loss: 157278093312.0000
Epoch 67/100
1064/1064 [==============================] - 1s 913us/step - loss: 1107793203
2.0000 - val_loss: 165541396480.0000
Epoch 68/100
1064/1064 [==============================] - 1s 999us/step - loss: 1106707148
8.0000 - val_loss: 156567846912.0000
Epoch 69/100
1064/1064 [==============================] - 1s 955us/step - loss: 1106243686
4.0000 - val_loss: 165919375360.0000
Epoch 70/100
1064/1064 [==============================] - 1s 999us/step - loss: 1104818483
2.0000 - val_loss: 167612366848.0000
Epoch 71/100
1064/1064 [==============================] - 1s 957us/step - loss: 1104126259
2.0000 - val_loss: 168193392640.0000
Epoch 72/100
1064/1064 [==============================] - 1s 951us/step - loss: 1103529369
```

```
6.0000 - val_loss: 166323830784.0000
Epoch 73/100
1064/1064 [==============================] - 1s 933us/step - loss: 1102580019
2.0000 - val_loss: 158965432320.0000
Epoch 74/100
1064/1064 [==============================] - 1s 941us/step - loss: 1102952243
2.0000 - val_loss: 186493321216.0000
Epoch 75/100
1064/1064 [==============================] - 1s 925us/step - loss: 1101570969
6.0000 - val_loss: 172662030336.0000
Epoch 76/100
1064/1064 [==============================] - 1s 944us/step - loss: 1100594585
6.0000 - val_loss: 169081503744.0000
Epoch 77/100
1064/1064 [==============================] - 1s 931us/step - loss: 1100408422
4.0000 - val_loss: 167957479424.0000
Epoch 78/100
1064/1064 [==============================] - 1s 998us/step - loss: 1098998681
6.0000 - val_loss: 170661019648.0000
Epoch 79/100
1064/1064 [==============================] - 1s 929us/step - loss: 1099627520
0.0000 - val_loss: 177441898496.0000
Epoch 80/100
1064/1064 [==============================] - 1s 961us/step - loss: 1098454937
6.0000 - val_loss: 170818420736.0000
Epoch 81/100
1064/1064 [==============================] - 1s 934us/step - loss: 1096620236
8.0000 - val_loss: 177505812480.0000
Epoch 82/100
1064/1064 [==============================] - 1s 937us/step - loss: 1096130252
8.0000 - val_loss: 180724482048.0000
Epoch 83/100
1064/1064 [==============================] - 1s 930us/step - loss: 1096347340
8.0000 - val_loss: 171737088000.0000
Epoch 84/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10963958784.
0000 - val_loss: 176354263040.0000
Epoch 85/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10943742976.
0000 - val_loss: 172222676992.0000
Epoch 86/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10945781760.
0000 - val_loss: 172448399360.0000
Epoch 87/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10952307712.
0000 - val_loss: 171793039360.0000
Epoch 88/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10936190976.
0000 - val_loss: 171370692608.0000
Epoch 89/100
1064/1064 [==============================] - 1s 988us/step - loss: 1093161984
0.0000 - val_loss: 178570379264.0000
Epoch 90/100
1064/1064 [==============================] - 1s 943us/step - loss: 1093070131
2.0000 - val_loss: 180039188480.0000
Epoch 91/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10923831296.
```

```
0000 - val_loss: 173050904576.0000
Epoch 92/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10921221120.
0000 - val_loss: 173433815040.0000
Epoch 93/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10912937984.
0000 - val_loss: 182240755712.0000
Epoch 94/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10911477760.
0000 - val_loss: 160751190016.0000
Epoch 95/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10908975104.
0000 - val_loss: 179574128640.0000
Epoch 96/100
1064/1064 [==============================] - 1s 951us/step - loss: 1089822105
6.0000 - val_loss: 171635769344.0000
Epoch 97/100
1064/1064 [==============================] - 1s 986us/step - loss: 1090994380
8.0000 - val_loss: 179873071104.0000
Epoch 98/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10891727872.
0000 - val_loss: 175017295872.0000
Epoch 99/100
1064/1064 [==============================] - 1s 979us/step - loss: 1088365670
4.0000 - val_loss: 184626118656.0000
Epoch 100/100
1064/1064 [==============================] - 1s 920us/step - loss: 1087673548
8.0000 - val_loss: 181681242112.0000
Epoch 1/100
WARNING:tensorflow:Layer dense_201 is casting an input tensor from dtype floa
t64 to the layer's dtype of float32, which is new behavior in TensorFlow 2.
The layer has dtype float32 because its dtype defaults to floatx.

If you intended to run this layer in float32, you can safely ignore this warn
ing. If in doubt, this warning is likely only an issue if you are porting a T
ensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backen
d.set_floatx('float64')`. To change just this layer, pass dtype='float64' to
the layer constructor. If you are the author of this layer, you can disable a
utocasting by passing autocast=False to the base Layer constructor.

1064/1064 [==============================] - 1s 1ms/step - loss: 62289788928.
0000 - val_loss: 70376579072.0000
Epoch 2/100
1064/1064 [==============================] - 1s 950us/step - loss: 1588780851
2.0000 - val_loss: 63667720192.0000
Epoch 3/100
1064/1064 [==============================] - 1s 1ms/step - loss: 14864551936.
0000 - val_loss: 49012969472.0000
Epoch 4/100
1064/1064 [==============================] - 1s 998us/step - loss: 1427249254
4.0000 - val_loss: 34828546048.0000
Epoch 5/100
1064/1064 [==============================] - 1s 946us/step - loss: 1387959091
2.0000 - val_loss: 25561982976.0000
Epoch 6/100
```

```
1064/1064 [==============================] - 1s 935us/step - loss: 1361306931
2.0000 - val_loss: 21620369408.0000
Epoch 7/100
1064/1064 [==============================] - 1s 981us/step - loss: 1340487680
0.0000 - val_loss: 19665973248.0000
Epoch 8/100
1064/1064 [==============================] - 1s 1ms/step - loss: 13255555072.
0000 - val_loss: 18210119680.0000
Epoch 9/100
1064/1064 [==============================] - 1s 1ms/step - loss: 13125841920.
0000 - val_loss: 16936879104.0000
Epoch 10/100
1064/1064 [==============================] - 1s 1ms/step - loss: 13011860480.
0000 - val_loss: 16012668928.0000
Epoch 11/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12909043712.
0000 - val_loss: 15375980544.0000
Epoch 12/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12827793408.
0000 - val_loss: 14999399424.0000
Epoch 13/100
1064/1064 [==============================] - 1s 996us/step - loss: 1272953241
6.0000 - val_loss: 15115657216.0000
Epoch 14/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12637230080.
0000 - val_loss: 15646139392.0000
Epoch 15/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12551930880.
0000 - val_loss: 16112450560.0000
Epoch 16/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12464005120.
0000 - val_loss: 17920747520.0000
Epoch 17/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12383760384.
0000 - val_loss: 17918416896.0000
Epoch 18/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12298292224.
0000 - val_loss: 19615791104.0000
Epoch 19/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12216274944.
0000 - val_loss: 20605872128.0000
Epoch 20/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12145609728.
0000 - val_loss: 23383119872.0000
Epoch 21/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12084891648.
0000 - val_loss: 26588641280.0000
Epoch 22/100
1064/1064 [==============================] - 1s 962us/step - loss: 1203093913
6.0000 - val_loss: 27163555840.0000
Epoch 23/100
1064/1064 [==============================] - 1s 969us/step - loss: 1197582848
0.0000 - val_loss: 30977994752.0000
Epoch 24/100
1064/1064 [==============================] - 1s 962us/step - loss: 1192024166
4.0000 - val_loss: 36875550720.0000
Epoch 25/100
```

```
1064/1064 [==============================] - 1s 986us/step - loss: 1187618304
0.0000 - val_loss: 39620284416.0000
Epoch 26/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11831590912.
0000 - val_loss: 42758918144.0000
Epoch 27/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11790742528.
0000 - val_loss: 47127789568.0000
Epoch 28/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11761858560.
0000 - val_loss: 48710787072.0000
Epoch 29/100
1064/1064 [==============================] - 1s 977us/step - loss: 1171698176
0.0000 - val_loss: 53284634624.0000
Epoch 30/100
1064/1064 [==============================] - 1s 937us/step - loss: 1169048780
8.0000 - val_loss: 59507449856.0000
Epoch 31/100
1064/1064 [==============================] - 1s 932us/step - loss: 1166957568
0.0000 - val_loss: 58791800832.0000
Epoch 32/100
1064/1064 [==============================] - 1s 916us/step - loss: 1163470540
8.0000 - val_loss: 62678032384.0000
Epoch 33/100
1064/1064 [==============================] - 1s 934us/step - loss: 1160460697
6.0000 - val_loss: 68381536256.0000
Epoch 34/100
1064/1064 [==============================] - 1s 940us/step - loss: 1158588518
4.0000 - val_loss: 68115058688.0000
Epoch 35/100
1064/1064 [==============================] - 1s 927us/step - loss: 1155269324
8.0000 - val_loss: 73336070144.0000
Epoch 36/100
1064/1064 [==============================] - 1s 907us/step - loss: 1153847808
0.0000 - val_loss: 76034949120.0000
Epoch 37/100
1064/1064 [==============================] - 1s 940us/step - loss: 1149833728
0.0000 - val_loss: 78107484160.0000
Epoch 38/100
1064/1064 [==============================] - 1s 923us/step - loss: 1148630835
2.0000 - val_loss: 84761452544.0000
Epoch 39/100
1064/1064 [==============================] - 1s 951us/step - loss: 1146600140
8.0000 - val_loss: 88288272384.0000
Epoch 40/100
1064/1064 [==============================] - 1s 948us/step - loss: 1144697548
8.0000 - val_loss: 89444974592.0000
Epoch 41/100
1064/1064 [==============================] - 1s 996us/step - loss: 1144165785
6.0000 - val_loss: 91242807296.0000
Epoch 42/100
1064/1064 [==============================] - 1s 944us/step - loss: 1141024153
6.0000 - val_loss: 96964050944.0000
Epoch 43/100
1064/1064 [==============================] - 1s 952us/step - loss: 1139181363
2.0000 - val_loss: 93910564864.0000
Epoch 44/100
```

```
1064/1064 [==============================] - 1s 952us/step - loss: 1137949081
6.0000 - val_loss: 97082728448.0000
Epoch 45/100
1064/1064 [==============================] - 1s 945us/step - loss: 1135540531
2.0000 - val_loss: 100840284160.0000
Epoch 46/100
1064/1064 [==============================] - 1s 941us/step - loss: 1133770547
2.0000 - val_loss: 103938809856.0000
Epoch 47/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11316634624.
0000 - val_loss: 102421127168.0000
Epoch 48/100
1064/1064 [==============================] - 1s 952us/step - loss: 1130193920
0.0000 - val_loss: 106512318464.0000
Epoch 49/100
1064/1064 [==============================] - 1s 944us/step - loss: 1128605388
8.0000 - val_loss: 111275081728.0000
Epoch 50/100
1064/1064 [==============================] - 1s 997us/step - loss: 1126355251
2.0000 - val_loss: 114010644480.0000
Epoch 51/100
1064/1064 [==============================] - 1s 950us/step - loss: 1124563558
4.0000 - val_loss: 113851392000.0000
Epoch 52/100
1064/1064 [==============================] - 1s 935us/step - loss: 1124086374
4.0000 - val_loss: 118151053312.0000
Epoch 53/100
1064/1064 [==============================] - 1s 925us/step - loss: 1121390592
0.0000 - val_loss: 117811118080.0000
Epoch 54/100
1064/1064 [==============================] - 1s 945us/step - loss: 1120821555
2.0000 - val_loss: 118854893568.0000
Epoch 55/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11181431808.
0000 - val_loss: 120079597568.0000
Epoch 56/100
1064/1064 [==============================] - 1s 989us/step - loss: 1118628864
0.0000 - val_loss: 119371964416.0000
Epoch 57/100
1064/1064 [==============================] - 1s 962us/step - loss: 1116357324
8.0000 - val_loss: 125009559552.0000
Epoch 58/100
1064/1064 [==============================] - 1s 925us/step - loss: 1115893657
6.0000 - val_loss: 124012789760.0000
Epoch 59/100
1064/1064 [==============================] - 1s 948us/step - loss: 1114005913
6.0000 - val_loss: 129136336896.0000
Epoch 60/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11137038336.
0000 - val_loss: 132154474496.0000
Epoch 61/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11115403264.
0000 - val_loss: 135619674112.0000
Epoch 62/100
1064/1064 [==============================] - 1s 976us/step - loss: 1111416422
4.0000 - val_loss: 140033310720.0000
Epoch 63/100
```

```
1064/1064 [==============================] - 1s 961us/step - loss: 1110684364
8.0000 - val_loss: 141754810368.0000
Epoch 64/100
1064/1064 [==============================] - 1s 938us/step - loss: 1108994867
2.0000 - val_loss: 143704735744.0000
Epoch 65/100
1064/1064 [==============================] - 1s 960us/step - loss: 1109361664
0.0000 - val_loss: 140287180800.0000
Epoch 66/100
1064/1064 [==============================] - 1s 931us/step - loss: 1106485145
6.0000 - val_loss: 146666225664.0000
Epoch 67/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11068278784.
0000 - val_loss: 149697495040.0000
Epoch 68/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11046841344.
0000 - val_loss: 153727385600.0000
Epoch 69/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11038736384.
0000 - val_loss: 151820877824.0000
Epoch 70/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11023304704.
0000 - val_loss: 157767761920.0000
Epoch 71/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11015998464.
0000 - val_loss: 160357859328.0000
Epoch 72/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10998503424.
0000 - val_loss: 167721992192.0000
Epoch 73/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10987281408.
0000 - val_loss: 160360202240.0000
Epoch 74/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10978409472.
0000 - val_loss: 168090550272.0000
Epoch 75/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10974670848.
0000 - val_loss: 166236422144.0000
Epoch 76/100
1064/1064 [==============================] - 1s 980us/step - loss: 1095244800
0.0000 - val_loss: 163370565632.0000
Epoch 77/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10940542976.
0000 - val_loss: 172763398144.0000
Epoch 78/100
1064/1064 [==============================] - 2s 1ms/step - loss: 10940640256.
0000 - val_loss: 180715274240.0000
Epoch 79/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10919339008.
0000 - val_loss: 169765257216.0000
Epoch 80/100
1064/1064 [==============================] - 2s 1ms/step - loss: 10915709952.
0000 - val_loss: 175688482816.0000
Epoch 81/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10901163008.
0000 - val_loss: 179552190464.0000
Epoch 82/100
```

```
1064/1064 [==============================] - 1s 1ms/step - loss: 10895591424.
0000 - val_loss: 186911113216.0000
Epoch 83/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10880135168.
0000 - val_loss: 192470384640.0000
Epoch 84/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10883444736.
0000 - val_loss: 191337168896.0000
Epoch 85/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10856797184.
0000 - val_loss: 204123865088.0000
Epoch 86/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10850757632.
0000 - val_loss: 196913774592.0000
Epoch 87/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10842009600.
0000 - val_loss: 202463756288.0000
Epoch 88/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10830552064.
0000 - val_loss: 213009416192.0000
Epoch 89/100
1064/1064 [==============================] - 1s 941us/step - loss: 1082110771
2.0000 - val_loss: 203969691648.0000
Epoch 90/100
1064/1064 [==============================] - 1s 844us/step - loss: 1080504832
0.0000 - val_loss: 213747204096.0000
Epoch 91/100
1064/1064 [==============================] - 1s 848us/step - loss: 1079242547
2.0000 - val_loss: 213800255488.0000
Epoch 92/100
1064/1064 [==============================] - 1s 839us/step - loss: 1078567833
6.0000 - val_loss: 225413234688.0000
Epoch 93/100
1064/1064 [==============================] - 1s 839us/step - loss: 1077664870
4.0000 - val_loss: 223108726784.0000
Epoch 94/100
1064/1064 [==============================] - 1s 841us/step - loss: 1077223526
4.0000 - val_loss: 229779619840.0000
Epoch 95/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10770356224.
0000 - val_loss: 218869268480.0000
Epoch 96/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10746604544.
0000 - val_loss: 221626253312.0000
Epoch 97/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10746406912.
0000 - val_loss: 239213133824.0000
Epoch 98/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10729841664.
0000 - val_loss: 234513940480.0000
Epoch 99/100
1064/1064 [==============================] - 1s 967us/step - loss: 1073040486
4.0000 - val_loss: 228751310848.0000
Epoch 100/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10720730112.
0000 - val_loss: 233431433216.0000
Epoch 1/100
```

```
WARNING:tensorflow:Layer dense_207 is casting an input tensor from dtype floa
t64 to the layer's dtype of float32, which is new behavior in TensorFlow 2.
The layer has dtype float32 because its dtype defaults to floatx.

If you intended to run this layer in float32, you can safely ignore this warn
ing. If in doubt, this warning is likely only an issue if you are porting a T
ensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backen
d.set_floatx('float64')`. To change just this layer, pass dtype='float64' to
the layer constructor. If you are the author of this layer, you can disable a
utocasting by passing autocast=False to the base Layer constructor.

1064/1064 [==============================] - 1s 1ms/step - loss: 51177070592.
0000 - val_loss: 74177011712.0000
Epoch 2/100
1064/1064 [==============================] - 1s 853us/step - loss: 1428323737
6.0000 - val_loss: 69672042496.0000
Epoch 3/100
1064/1064 [==============================] - 1s 926us/step - loss: 1336940851
2.0000 - val_loss: 56467451904.0000
Epoch 4/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12861970432.
0000 - val_loss: 45665423360.0000
Epoch 5/100
1064/1064 [==============================] - 1s 948us/step - loss: 1256782131
2.0000 - val_loss: 37454061568.0000
Epoch 6/100
1064/1064 [==============================] - 1s 995us/step - loss: 1236547686
4.0000 - val_loss: 30380525568.0000
Epoch 7/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12212695040.
0000 - val_loss: 25432745984.0000
Epoch 8/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12074799104.
0000 - val_loss: 23045345280.0000
Epoch 9/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11970360320.
0000 - val_loss: 20920295424.0000
Epoch 10/100
1064/1064 [==============================] - 1s 844us/step - loss: 1187515596
8.0000 - val_loss: 20664121344.0000
Epoch 11/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11783093248.
0000 - val_loss: 20590376960.0000
Epoch 12/100
1064/1064 [==============================] - 1s 962us/step - loss: 1169670451
2.0000 - val_loss: 21067079680.0000
Epoch 13/100
1064/1064 [==============================] - 1s 946us/step - loss: 1161231462
4.0000 - val_loss: 22508414976.0000
Epoch 14/100
1064/1064 [==============================] - 1s 849us/step - loss: 1154938163
2.0000 - val_loss: 24114589696.0000
Epoch 15/100
1064/1064 [==============================] - 1s 886us/step - loss: 1148316467
2.0000 - val_loss: 26227580928.0000
```

```
Epoch 16/100
1064/1064 [==============================] - 1s 957us/step - loss: 1139927449
6.0000 - val_loss: 29860100096.0000
Epoch 17/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11319311360.
0000 - val_loss: 32793307136.0000
Epoch 18/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11254252544.
0000 - val_loss: 37080408064.0000
Epoch 19/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11187047424.
0000 - val_loss: 38107807744.0000
Epoch 20/100
1064/1064 [==============================] - 1s 862us/step - loss: 1113721241
6.0000 - val_loss: 46034055168.0000
Epoch 21/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11091007488.
0000 - val_loss: 48878137344.0000
Epoch 22/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11050268672.
0000 - val_loss: 52279418880.0000
Epoch 23/100
1064/1064 [==============================] - 1s 860us/step - loss: 1101799321
6.0000 - val_loss: 55378542592.0000
Epoch 24/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10971959296.
0000 - val_loss: 61234270208.0000
Epoch 25/100
1064/1064 [==============================] - 1s 995us/step - loss: 1095362048
0.0000 - val_loss: 63527272448.0000
Epoch 26/100
1064/1064 [==============================] - 1s 909us/step - loss: 1092612812
8.0000 - val_loss: 71169925120.0000
Epoch 27/100
1064/1064 [==============================] - 1s 906us/step - loss: 1089973862
4.0000 - val_loss: 75630854144.0000
Epoch 28/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10876169216.
0000 - val_loss: 80060014592.0000
Epoch 29/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10857957376.
0000 - val_loss: 85873623040.0000
Epoch 30/100
1064/1064 [==============================] - 2s 1ms/step - loss: 10832067584.
0000 - val_loss: 87251386368.0000
Epoch 31/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10822371328.
0000 - val_loss: 96902537216.0000
Epoch 32/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10801880064.
0000 - val_loss: 95837995008.0000
Epoch 33/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10776414208.
0000 - val_loss: 100437368832.0000
Epoch 34/100
1064/1064 [==============================] - 2s 1ms/step - loss: 10772210688.
0000 - val_loss: 109712334848.0000
```

```
Epoch 35/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10749969408.
0000 - val_loss: 111488122880.0000
Epoch 36/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10736436224.
0000 - val_loss: 119324205056.0000
Epoch 37/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10716187648.
0000 - val_loss: 121201262592.0000
Epoch 38/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10701928448.
0000 - val_loss: 123157323776.0000
Epoch 39/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10695813120.
0000 - val_loss: 134809722880.0000
Epoch 40/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10679128064.
0000 - val_loss: 134714564608.0000
Epoch 41/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10670359552.
0000 - val_loss: 145471258624.0000
Epoch 42/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10655372288.
0000 - val_loss: 150742614016.0000
Epoch 43/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10648139776.
0000 - val_loss: 151595450368.0000
Epoch 44/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10641873920.
0000 - val_loss: 152461410304.0000
Epoch 45/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10627785728.
0000 - val_loss: 163240198144.0000
Epoch 46/100
1064/1064 [==============================] - 1s 964us/step - loss: 1060611686
4.0000 - val_loss: 174074019840.0000
Epoch 47/100
1064/1064 [==============================] - 1s 983us/step - loss: 1059535974
4.0000 - val_loss: 176755507200.0000
Epoch 48/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10585776128.
0000 - val_loss: 191276630016.0000
Epoch 49/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10580020224.
0000 - val_loss: 191630737408.0000
Epoch 50/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10564880384.
0000 - val_loss: 194448293888.0000
Epoch 51/100
1064/1064 [==============================] - 1s 949us/step - loss: 1055251865
6.0000 - val_loss: 208655187968.0000
Epoch 52/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10548687872.
0000 - val_loss: 219872837632.0000
Epoch 53/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10537368576.
0000 - val_loss: 218038435840.0000
```

```
Epoch 54/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10531240960.
0000 - val_loss: 235487805440.0000
Epoch 55/100
1064/1064 [==============================] - 2s 2ms/step - loss: 10508110848.
0000 - val_loss: 237037699072.0000
Epoch 56/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10509708288.
0000 - val_loss: 250843250688.0000
Epoch 57/100
1064/1064 [==============================] - 2s 2ms/step - loss: 10500177920.
0000 - val_loss: 248111071232.0000
Epoch 58/100
1064/1064 [==============================] - 2s 2ms/step - loss: 10491874304.
0000 - val_loss: 267607752704.0000
Epoch 59/100
1064/1064 [==============================] - 2s 2ms/step - loss: 10480156672.
0000 - val_loss: 270016233472.0000
Epoch 60/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10468754432.
0000 - val_loss: 280760483840.0000
Epoch 61/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10471188480.
0000 - val_loss: 271767945216.0000
Epoch 62/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10456879104.
0000 - val_loss: 282641825792.0000
Epoch 63/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10450588672.
0000 - val_loss: 296037154816.0000
Epoch 64/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10433973248.
0000 - val_loss: 302672707584.0000
Epoch 65/100
1064/1064 [==============================] - 2s 1ms/step - loss: 10436939776.
0000 - val_loss: 306326011904.0000
Epoch 66/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10429043712.
0000 - val_loss: 334498037760.0000
Epoch 67/100
1064/1064 [==============================] - 1s 976us/step - loss: 1043011174
4.0000 - val_loss: 320677642240.0000
Epoch 68/100
1064/1064 [==============================] - 1s 930us/step - loss: 1041325977
6.0000 - val_loss: 344276893696.0000
Epoch 69/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10398188544.
0000 - val_loss: 343738777600.0000
Epoch 70/100
1064/1064 [==============================] - 2s 2ms/step - loss: 10396769280.
0000 - val_loss: 355572023296.0000
Epoch 71/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10388610048.
0000 - val_loss: 349689118720.0000
Epoch 72/100
1064/1064 [==============================] - 2s 2ms/step - loss: 10392200192.
0000 - val_loss: 364222644224.0000
```

```
Epoch 73/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10372092928.
0000 - val_loss: 367567372288.0000
Epoch 74/100
1064/1064 [==============================] - 2s 1ms/step - loss: 10371229696.
0000 - val_loss: 391460028416.0000
Epoch 75/100
1064/1064 [==============================] - 2s 1ms/step - loss: 10356147200.
0000 - val_loss: 405893808128.0000
Epoch 76/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10355487744.
0000 - val_loss: 405934768128.0000
Epoch 77/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10349068288.
0000 - val_loss: 389220925440.0000
Epoch 78/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10337674240.
0000 - val_loss: 411902279680.0000
Epoch 79/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10343550976.
0000 - val_loss: 417799143424.0000
Epoch 80/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10325937152.
0000 - val_loss: 439828054016.0000
Epoch 81/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10312760320.
0000 - val_loss: 446411833344.0000
Epoch 82/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10325401600.
0000 - val_loss: 465100570624.0000
Epoch 83/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10315737088.
0000 - val_loss: 479901089792.0000
Epoch 84/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10296687616.
0000 - val_loss: 475261796352.0000
Epoch 85/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10294617088.
0000 - val_loss: 485032919040.0000
Epoch 86/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10292023296.
0000 - val_loss: 497918214144.0000
Epoch 87/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10278509568.
0000 - val_loss: 486608961536.0000
Epoch 88/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10271782912.
0000 - val_loss: 504391368704.0000
Epoch 89/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10269554688.
0000 - val_loss: 537477021696.0000
Epoch 90/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10253815808.
0000 - val_loss: 525106872320.0000
Epoch 91/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10255426560.
0000 - val_loss: 540535750656.0000
```

```
Epoch 92/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10252565504.
0000 - val_loss: 550126747648.0000
Epoch 93/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10244366336.
0000 - val_loss: 561820663808.0000
Epoch 94/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10227173376.
0000 - val_loss: 570232733696.0000
Epoch 95/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10221932544.
0000 - val_loss: 558144356352.0000
Epoch 96/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10214896640.
0000 - val_loss: 578575925248.0000
Epoch 97/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10212174848.
0000 - val_loss: 591160672256.0000
Epoch 98/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10195898368.
0000 - val_loss: 558976794624.0000
Epoch 99/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10201959424.
0000 - val_loss: 596358791168.0000
Epoch 100/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10187860992.
0000 - val_loss: 608555827200.0000
Epoch 1/100
WARNING:tensorflow:Layer dense_213 is casting an input tensor from dtype floa
t64 to the layer's dtype of float32, which is new behavior in TensorFlow 2.
The layer has dtype float32 because its dtype defaults to floatx.

If you intended to run this layer in float32, you can safely ignore this warn
ing. If in doubt, this warning is likely only an issue if you are porting a T
ensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backen
d.set_floatx('float64')`. To change just this layer, pass dtype='float64' to
the layer constructor. If you are the author of this layer, you can disable a
utocasting by passing autocast=False to the base Layer constructor.

1064/1064 [==============================] - 1s 1ms/step - loss: 52804882432.
0000 - val_loss: 78227824640.0000
Epoch 2/100
1064/1064 [==============================] - 1s 1ms/step - loss: 14510166016.
0000 - val_loss: 64559190016.0000
Epoch 3/100
1064/1064 [==============================] - 1s 1ms/step - loss: 13600645120.
0000 - val_loss: 53285236736.0000
Epoch 4/100
1064/1064 [==============================] - 1s 1ms/step - loss: 13153610752.
0000 - val_loss: 42483580928.0000
Epoch 5/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12875628544.
0000 - val_loss: 34546982912.0000
Epoch 6/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12669473792.
```

```
0000 - val_loss: 27512301568.0000
Epoch 7/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12496987136.
0000 - val_loss: 23906695168.0000
Epoch 8/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12335073280.
0000 - val_loss: 20943790080.0000
Epoch 9/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12188302336.
0000 - val_loss: 18758705152.0000
Epoch 10/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12049709056.
0000 - val_loss: 18043191296.0000
Epoch 11/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11925315584.
0000 - val_loss: 18317592576.0000
Epoch 12/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11811689472.
0000 - val_loss: 20543193088.0000
Epoch 13/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11692709888.
0000 - val_loss: 24080963584.0000
Epoch 14/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11600818176.
0000 - val_loss: 27910600704.0000
Epoch 15/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11523555328.
0000 - val_loss: 31870795776.0000
Epoch 16/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11435475968.
0000 - val_loss: 35923501056.0000
Epoch 17/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11361034240.
0000 - val_loss: 41416781824.0000
Epoch 18/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11297926144.
0000 - val_loss: 47593484288.0000
Epoch 19/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11243005952.
0000 - val_loss: 50451181568.0000
Epoch 20/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11199517696.
0000 - val_loss: 52829917184.0000
Epoch 21/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11164800000.
0000 - val_loss: 58309754880.0000
Epoch 22/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11129650176.
0000 - val_loss: 66700656640.0000
Epoch 23/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11099537408.
0000 - val_loss: 68690198528.0000
Epoch 24/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11064796160.
0000 - val_loss: 79303761920.0000
Epoch 25/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11053713408.
```

```
0000 - val_loss: 82046369792.0000
Epoch 26/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11017798656.
0000 - val_loss: 95121457152.0000
Epoch 27/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11005619200.
0000 - val_loss: 101813297152.0000
Epoch 28/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10980543488.
0000 - val_loss: 103951319040.0000
Epoch 29/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10958560256.
0000 - val_loss: 106857046016.0000
Epoch 30/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10939749376.
0000 - val_loss: 112634019840.0000
Epoch 31/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10930465792.
0000 - val_loss: 124102344704.0000
Epoch 32/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10908359680.
0000 - val_loss: 131497459712.0000
Epoch 33/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10896149504.
0000 - val_loss: 133493981184.0000
Epoch 34/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10880374784.
0000 - val_loss: 138806689792.0000
Epoch 35/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10862115840.
0000 - val_loss: 150109536256.0000
Epoch 36/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10842673152.
0000 - val_loss: 147896745984.0000
Epoch 37/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10842678272.
0000 - val_loss: 168894808064.0000
Epoch 38/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10820288512.
0000 - val_loss: 163860660224.0000
Epoch 39/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10814566400.
0000 - val_loss: 187093663744.0000
Epoch 40/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10802660352.
0000 - val_loss: 190387912704.0000
Epoch 41/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10781261824.
0000 - val_loss: 194609299456.0000
Epoch 42/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10783112192.
0000 - val_loss: 208867344384.0000
Epoch 43/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10767872000.
0000 - val_loss: 199786266624.0000
Epoch 44/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10756458496.
```

```
0000 - val_loss: 216561614848.0000
Epoch 45/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10743486464.
0000 - val_loss: 222908153856.0000
Epoch 46/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10730969088.
0000 - val_loss: 233610313728.0000
Epoch 47/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10721347584.
0000 - val_loss: 230189760512.0000
Epoch 48/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10709120000.
0000 - val_loss: 244833222656.0000
Epoch 49/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10703865856.
0000 - val_loss: 250251788288.0000
Epoch 50/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10684904448.
0000 - val_loss: 250107740160.0000
Epoch 51/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10679472128.
0000 - val_loss: 268708134912.0000
Epoch 52/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10669154304.
0000 - val_loss: 277947383808.0000
Epoch 53/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10662703104.
0000 - val_loss: 278993338368.0000
Epoch 54/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10651210752.
0000 - val_loss: 292859150336.0000
Epoch 55/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10650940416.
0000 - val_loss: 288661962752.0000
Epoch 56/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10635681792.
0000 - val_loss: 302665367552.0000
Epoch 57/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10621601792.
0000 - val_loss: 315465105408.0000
Epoch 58/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10619498496.
0000 - val_loss: 316597108736.0000
Epoch 59/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10613467136.
0000 - val_loss: 311726407680.0000
Epoch 60/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10610979840.
0000 - val_loss: 321982562304.0000
Epoch 61/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10594644992.
0000 - val_loss: 317920215040.0000
Epoch 62/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10588823552.
0000 - val_loss: 345081774080.0000
Epoch 63/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10578646016.
```

```
0000 - val_loss: 339735117824.0000
Epoch 64/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10575413248.
0000 - val_loss: 338721439744.0000
Epoch 65/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10570108928.
0000 - val_loss: 335460728832.0000
Epoch 66/100
1064/1064 [==============================] - 2s 1ms/step - loss: 10562922496.
0000 - val_loss: 350186504192.0000
Epoch 67/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10548627456.
0000 - val_loss: 348144140288.0000
Epoch 68/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10546382848.
0000 - val_loss: 356937859072.0000
Epoch 69/100
1064/1064 [==============================] - 2s 2ms/step - loss: 10543796224.
0000 - val_loss: 367783411712.0000
Epoch 70/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10546099200.
0000 - val_loss: 359778123776.0000
Epoch 71/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10529487872.
0000 - val_loss: 383588007936.0000
Epoch 72/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10531622912.
0000 - val_loss: 382507515904.0000
Epoch 73/100
1064/1064 [==============================] - 2s 2ms/step - loss: 10518223872.
0000 - val_loss: 376541970432.0000
Epoch 74/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10508196864.
0000 - val_loss: 368833298432.0000
Epoch 75/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10500313088.
0000 - val_loss: 387877732352.0000
Epoch 76/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10492165120.
0000 - val_loss: 379077427200.0000
Epoch 77/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10490233856.
0000 - val_loss: 381414473728.0000
Epoch 78/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10485906432.
0000 - val_loss: 413450305536.0000
Epoch 79/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10486837248.
0000 - val_loss: 410498170880.0000
Epoch 80/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10481931264.
0000 - val_loss: 406376284160.0000
Epoch 81/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10468699136.
0000 - val_loss: 408836276224.0000
Epoch 82/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10469178368.
```

```
0000 - val_loss: 399053783040.0000
Epoch 83/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10462958592.
0000 - val_loss: 413000892416.0000
Epoch 84/100
1064/1064 [==============================] - 1s 1000us/step - loss: 104485099
52.0000 - val_loss: 395185520640.0000
Epoch 85/100
1064/1064 [==============================] - 1s 995us/step - loss: 1044077875
2.0000 - val_loss: 425550610432.0000
Epoch 86/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10439435264.
0000 - val_loss: 409554780160.0000
Epoch 87/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10436589568.
0000 - val_loss: 415841157120.0000
Epoch 88/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10423269376.
0000 - val_loss: 425639706624.0000
Epoch 89/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10413133824.
0000 - val_loss: 424974254080.0000
Epoch 90/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10409527296.
0000 - val_loss: 439467311104.0000
Epoch 91/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10399724544.
0000 - val_loss: 444355215360.0000
Epoch 92/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10399563776.
0000 - val_loss: 453504335872.0000
Epoch 93/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10389937152.
0000 - val_loss: 457974054912.0000
Epoch 94/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10395654144.
0000 - val_loss: 447729303552.0000
Epoch 95/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10375875584.
0000 - val_loss: 448078151680.0000
Epoch 96/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10379679744.
0000 - val_loss: 460679512064.0000
Epoch 97/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10364184576.
0000 - val_loss: 458027794432.0000
Epoch 98/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10366939136.
0000 - val_loss: 450805596160.0000
Epoch 99/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10357605376.
0000 - val_loss: 452688248832.0000
Epoch 100/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10350799872.
0000 - val_loss: 458947657728.0000
Epoch 1/100
WARNING:tensorflow:Layer dense_219 is casting an input tensor from dtype floa
```

```
t64 to the layer's dtype of float32, which is new behavior in TensorFlow 2.
The layer has dtype float32 because its dtype defaults to floatx.

If you intended to run this layer in float32, you can safely ignore this warn
ing. If in doubt, this warning is likely only an issue if you are porting a T
ensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backen
d.set_floatx('float64')`. To change just this layer, pass dtype='float64' to
the layer constructor. If you are the author of this layer, you can disable a
utocasting by passing autocast=False to the base Layer constructor.

1064/1064 [==============================] - 1s 1ms/step - loss: 44107509760.
0000 - val_loss: 246558523392.0000
Epoch 2/100
1064/1064 [==============================] - 1s 1ms/step - loss: 13675321344.
0000 - val_loss: 231608909824.0000
Epoch 3/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12934937600.
0000 - val_loss: 206993113088.0000
Epoch 4/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12494206976.
0000 - val_loss: 176956964864.0000
Epoch 5/100
1064/1064 [==============================] - 1s 1ms/step - loss: 12199048192.
0000 - val_loss: 152394317824.0000
Epoch 6/100
1064/1064 [==============================] - 1s 999us/step - loss: 1200727040
0.0000 - val_loss: 128626999296.0000
Epoch 7/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11865816064.
0000 - val_loss: 113516535808.0000
Epoch 8/100
1064/1064 [==============================] - 1s 987us/step - loss: 1173864038
4.0000 - val_loss: 95351996416.0000
Epoch 9/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11633352704.
0000 - val_loss: 80817594368.0000
Epoch 10/100
1064/1064 [==============================] - 1s 980us/step - loss: 1154465792
0.0000 - val_loss: 69388959744.0000
Epoch 11/100
1064/1064 [==============================] - 1s 997us/step - loss: 1144883814
4.0000 - val_loss: 58748506112.0000
Epoch 12/100
1064/1064 [==============================] - 1s 998us/step - loss: 1137875968
0.0000 - val_loss: 52869431296.0000
Epoch 13/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11302802432.
0000 - val_loss: 46003269632.0000
Epoch 14/100
1064/1064 [==============================] - 1s 977us/step - loss: 1124678144
0.0000 - val_loss: 40639119360.0000
Epoch 15/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11179579392.
0000 - val_loss: 37125955584.0000
Epoch 16/100
```

```
1064/1064 [==============================] - 1s 1ms/step - loss: 11115315200.
0000 - val_loss: 35525783552.0000
Epoch 17/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11068755968.
0000 - val_loss: 36414758912.0000
Epoch 18/100
1064/1064 [==============================] - 1s 1ms/step - loss: 11030546432.
0000 - val_loss: 38967074816.0000
Epoch 19/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10979784704.
0000 - val_loss: 42079363072.0000
Epoch 20/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10944314368.
0000 - val_loss: 50060660736.0000
Epoch 21/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10906512384.
0000 - val_loss: 57213681664.0000
Epoch 22/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10873812992.
0000 - val_loss: 64026595328.0000
Epoch 23/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10830812160.
0000 - val_loss: 76335153152.0000
Epoch 24/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10802151424.
0000 - val_loss: 92551323648.0000
Epoch 25/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10771620864.
0000 - val_loss: 111146680320.0000
Epoch 26/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10733763584.
0000 - val_loss: 130468995072.0000
Epoch 27/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10705502208.
0000 - val_loss: 145163632640.0000
Epoch 28/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10688517120.
0000 - val_loss: 169115697152.0000
Epoch 29/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10659399680.
0000 - val_loss: 193496825856.0000
Epoch 30/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10628162560.
0000 - val_loss: 208679436288.0000
Epoch 31/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10608310272.
0000 - val_loss: 227669786624.0000
Epoch 32/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10596584448.
0000 - val_loss: 254970298368.0000
Epoch 33/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10578138112.
0000 - val_loss: 260336943104.0000
Epoch 34/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10564956160.
0000 - val_loss: 290684010496.0000
Epoch 35/100
```

```
1064/1064 [==============================] - 1s 1ms/step - loss: 10538618880.
0000 - val_loss: 298213736448.0000
Epoch 36/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10525900800.
0000 - val_loss: 324050812928.0000
Epoch 37/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10503877632.
0000 - val_loss: 355600334848.0000
Epoch 38/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10488233984.
0000 - val_loss: 372204503040.0000
Epoch 39/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10472787968.
0000 - val_loss: 421584666624.0000
Epoch 40/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10454843392.
0000 - val_loss: 432481533952.0000
Epoch 41/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10433422336.
0000 - val_loss: 457991323648.0000
Epoch 42/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10429256704.
0000 - val_loss: 479438864384.0000
Epoch 43/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10402810880.
0000 - val_loss: 504081973248.0000
Epoch 44/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10399455232.
0000 - val_loss: 535234117632.0000
Epoch 45/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10381101056.
0000 - val_loss: 581636718592.0000
Epoch 46/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10377578496.
0000 - val_loss: 599722557440.0000
Epoch 47/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10355472384.
0000 - val_loss: 624986423296.0000
Epoch 48/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10355671040.
0000 - val_loss: 639670812672.0000
Epoch 49/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10334982144.
0000 - val_loss: 680566259712.0000
Epoch 50/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10327247872.
0000 - val_loss: 680477065216.0000
Epoch 51/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10316622848.
0000 - val_loss: 744223080448.0000
Epoch 52/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10296300544.
0000 - val_loss: 766195138560.0000
Epoch 53/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10280664064.
0000 - val_loss: 789168390144.0000
Epoch 54/100
```

```
1064/1064 [==============================] - 1s 1ms/step - loss: 10274975744.
0000 - val_loss: 802609364992.0000
Epoch 55/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10261122048.
0000 - val_loss: 858181861376.0000
Epoch 56/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10241506304.
0000 - val_loss: 925615718400.0000
Epoch 57/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10236267520.
0000 - val_loss: 957708042240.0000
Epoch 58/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10230801408.
0000 - val_loss: 973874266112.0000
Epoch 59/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10219513856.
0000 - val_loss: 963026944000.0000
Epoch 60/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10203011072.
0000 - val_loss: 1027118596096.0000
Epoch 61/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10190880768.
0000 - val_loss: 1071344189440.0000
Epoch 62/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10185322496.
0000 - val_loss: 1091426582528.0000
Epoch 63/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10166539264.
0000 - val_loss: 1092574052352.0000
Epoch 64/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10168039424.
0000 - val_loss: 1157490671616.0000
Epoch 65/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10144572416.
0000 - val_loss: 1227085316096.0000
Epoch 66/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10135577600.
0000 - val_loss: 1267857489920.0000
Epoch 67/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10128685056.
0000 - val_loss: 1263626878976.0000
Epoch 68/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10116226048.
0000 - val_loss: 1323217846272.0000
Epoch 69/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10104707072.
0000 - val_loss: 1343295979520.0000
Epoch 70/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10080723968.
0000 - val_loss: 1369166184448.0000
Epoch 71/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10083406848.
0000 - val_loss: 1350920437760.0000
Epoch 72/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10070154240.
0000 - val_loss: 1387816943616.0000
Epoch 73/100
```

```
1064/1064 [==============================] - 1s 1ms/step - loss: 10063761408.
0000 - val_loss: 1491531202560.0000
Epoch 74/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10059343872.
0000 - val_loss: 1523394936832.0000
Epoch 75/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10038334464.
0000 - val_loss: 1519782068224.0000
Epoch 76/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10034588672.
0000 - val_loss: 1550830665728.0000
Epoch 77/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10022955008.
0000 - val_loss: 1564958392320.0000
Epoch 78/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10005516288.
0000 - val_loss: 1637659443200.0000
Epoch 79/100
1064/1064 [==============================] - 1s 1ms/step - loss: 10000785408.
0000 - val_loss: 1684872364032.0000
Epoch 80/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9999033344.0
000 - val_loss: 1666253455360.0000
Epoch 81/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9984743424.0
000 - val_loss: 1682183159808.0000
Epoch 82/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9978121216.0
000 - val_loss: 1676613779456.0000
Epoch 83/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9973076992.0
000 - val_loss: 1712729489408.0000
Epoch 84/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9956719616.0
000 - val_loss: 1747380076544.0000
Epoch 85/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9947816960.0
000 - val_loss: 1768065073152.0000
Epoch 86/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9945443328.0
000 - val_loss: 1752163549184.0000
Epoch 87/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9931223040.0
000 - val_loss: 1860119035904.0000
Epoch 88/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9934592000.0
000 - val_loss: 1811474808832.0000
Epoch 89/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9925043200.0
000 - val_loss: 1792547356672.0000
Epoch 90/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9924937728.0
000 - val_loss: 1804008947712.0000
Epoch 91/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9909208064.0
000 - val_loss: 1927947747328.0000
Epoch 92/100
```

```
1064/1064 [==============================] - 1s 1ms/step - loss: 9910005760.0
000 - val_loss: 1943749918720.0000
Epoch 93/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9895170048.0
000 - val_loss: 1927978549248.0000
Epoch 94/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9890558976.0
000 - val_loss: 1884746153984.0000
Epoch 95/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9894507520.0
000 - val_loss: 1958110560256.0000
Epoch 96/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9892077568.0
000 - val_loss: 1945645481984.0000
Epoch 97/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9874844672.0
000 - val_loss: 1927074807808.0000
Epoch 98/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9866102784.0
000 - val_loss: 1998697267200.0000
Epoch 99/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9879014400.0
000 - val_loss: 1949909647360.0000
Epoch 100/100
1064/1064 [==============================] - 1s 1ms/step - loss: 9858245632.0
000 - val_loss: 1957814468608.0000
```

In [107]:
```
val_loss_mean
```

Out[107]:
```
[array([1.07506118e+11, 9.87117265e+10, 8.35994223e+10, 6.73364943e+10,
        5.52976949e+10, 4.55608799e+10, 3.98519906e+10, 3.46957828e+10,
        3.08427577e+10, 2.84811944e+10, 2.67739091e+10, 2.66929506e+10,
        2.75846599e+10, 2.82435455e+10, 2.99277679e+10, 3.30499727e+10,
        3.57106737e+10, 4.05252092e+10, 4.26295800e+10, 4.80619532e+10,
        5.29017442e+10, 5.73219672e+10, 6.25592263e+10, 7.21032602e+10,
        7.83096930e+10, 8.67543572e+10, 9.39615011e+10, 1.01085335e+11,
        1.10731547e+11, 1.16420218e+11, 1.24879899e+11, 1.33332523e+11,
        1.37050670e+11, 1.45734580e+11, 1.51457541e+11, 1.59649840e+11,
        1.71986919e+11, 1.76069991e+11, 1.95710481e+11, 1.97606715e+11,
        2.05405268e+11, 2.15863912e+11, 2.17039841e+11, 2.28331495e+11,
        2.42323228e+11, 2.50959513e+11, 2.57105197e+11, 2.65604973e+11,
        2.78319828e+11, 2.78525095e+11, 2.95308442e+11, 3.04929646e+11,
        3.10339040e+11, 3.17640055e+11, 3.31361989e+11, 3.48479581e+11,
        3.58011915e+11, 3.67106825e+11, 3.65354587e+11, 3.82460900e+11,
        3.90739234e+11, 4.03423622e+11, 4.06249642e+11, 4.21235147e+11,
        4.35352270e+11, 4.51297270e+11, 4.49537511e+11, 4.66945566e+11,
        4.74511684e+11, 4.81979292e+11, 4.82549763e+11, 4.93718585e+11,
        5.10993236e+11, 5.27654427e+11, 5.30490412e+11, 5.33658986e+11,
        5.35262934e+11, 5.62877664e+11, 5.72075367e+11, 5.71792939e+11,
        5.78897854e+11, 5.81680746e+11, 5.93967789e+11, 5.97103765e+11,
        6.10999029e+11, 6.05799743e+11, 6.27365190e+11, 6.25177199e+11,
        6.27507741e+11, 6.32473905e+11, 6.59937975e+11, 6.69245610e+11,
        6.70624550e+11, 6.58647800e+11, 6.72555293e+11, 6.75632588e+11,
        6.79069896e+11, 6.83602179e+11, 6.82466823e+11, 6.88086126e+11])]
```

In [115]:
```
# find the best epoch
best_epoch = np.argmin(val_loss_mean[0])+1
best_epoch
```

Out[115]: 12

In [117]:
```
# Train the model again on the entire testing set with the best parameters.
clt_nn = creat_model(5)
clt_nn.fit(x=X_train,y=y_train,
          validation_data=(X_test,y_test),
          batch_size=256,epochs=best_epoch)
```

```
Epoch 1/12
1330/1330 [==============================] - 1s 931us/step - loss: 4859992473
6.0000 - val_loss: 215038197760.0000
Epoch 2/12
1330/1330 [==============================] - 1s 868us/step - loss: 1478702284
8.0000 - val_loss: 204508037120.0000
Epoch 3/12
1330/1330 [==============================] - 1s 869us/step - loss: 1387418419
2.0000 - val_loss: 178344050688.0000
Epoch 4/12
1330/1330 [==============================] - 1s 861us/step - loss: 1338677760
0.0000 - val_loss: 145075879936.0000
Epoch 5/12
1330/1330 [==============================] - 1s 849us/step - loss: 1309770035
2.0000 - val_loss: 115701194752.0000
Epoch 6/12
1330/1330 [==============================] - 1s 849us/step - loss: 1288504320
0.0000 - val_loss: 89730818048.0000
Epoch 7/12
1330/1330 [==============================] - 1s 863us/step - loss: 1272692940
8.0000 - val_loss: 66775109632.0000
Epoch 8/12
1330/1330 [==============================] - 1s 845us/step - loss: 1259812454
4.0000 - val_loss: 50425892864.0000
Epoch 9/12
1330/1330 [==============================] - 1s 844us/step - loss: 1249677721
6.0000 - val_loss: 40979849216.0000
Epoch 10/12
1330/1330 [==============================] - 1s 859us/step - loss: 1241415577
6.0000 - val_loss: 36094009344.0000
Epoch 11/12
1330/1330 [==============================] - 1s 850us/step - loss: 1233134592
0.0000 - val_loss: 34952568832.0000
Epoch 12/12
1330/1330 [==============================] - 1s 850us/step - loss: 1227023257
6.0000 - val_loss: 40774336512.0000
```

Out[117]: <tensorflow.python.keras.callbacks.History at 0x1c867126a08>

2.2.3 Make Prediction on Testing Set

In [118]:
```
y_pred_nn = clt_nn.predict(X_test)
```

2.3 Decision Trees

2.3.1 Seperate Dataset

In [119]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
```

2.3.2 Train Model

In [120]:
```
params_grid = [{'max_depth':[5,10,15,20,25]}]

clf_dt_grid = GridSearchCV(DecisionTreeRegressor(), params_grid, cv=5)
clf_dt_grid.fit(X_train, y_train)
```

Out[120]:
```
GridSearchCV(cv=5, error_score=nan,
             estimator=DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse',
                                             max_depth=None, max_features=Non
e,
                                             max_leaf_nodes=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             presort='deprecated',
                                             random_state=None,
                                             splitter='best'),
             iid='deprecated', n_jobs=None,
             param_grid=[{'max_depth': [5, 10, 15, 20, 25]}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

In [121]:
```
clf_dt = clf_dt_grid.best_estimator_
print('Best max_depth:',clf_dt_grid.best_estimator_.max_depth,"\n")
```

```
Best max_depth: 10
```

2.3.3 Make Prediction on Testing Set

In [122]:
```
y_pred_dt = clf_dt.predict(X_test)
```

2.4 Random Forests

In [123]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
```

2.3.2 Train Model

In [124]:
```
params_grid = [{'max_depth':[5,10,15,20,25]}]

clf_rf_grid = GridSearchCV(RandomForestRegressor(), params_grid, cv=5)
clf_rf_grid.fit(X_train, y_train)
```

```
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
```

```
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
```

```
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:515: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\kaike\anaconda3\lib\site-packages\sklearn\model_selection\_search.p
y:739: DataConversionWarning: A column-vector y was passed when a 1d array wa
s expected. Please change the shape of y to (n_samples,), for example using r
avel().
  self.best_estimator_.fit(X, y, **fit_params)
```

Out[124]:
```
GridSearchCV(cv=5, error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=100, n_jobs=None,
                                             oob_score=False, random_state=No
ne,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid=[{'max_depth': [5, 10, 15, 20, 25]}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

In [125]:
```
clf_rf = clf_rf_grid.best_estimator_
print('Best max_depth:',clf_rf_grid.best_estimator_.max_depth,"\n")
```

```
Best max_depth: 25
```

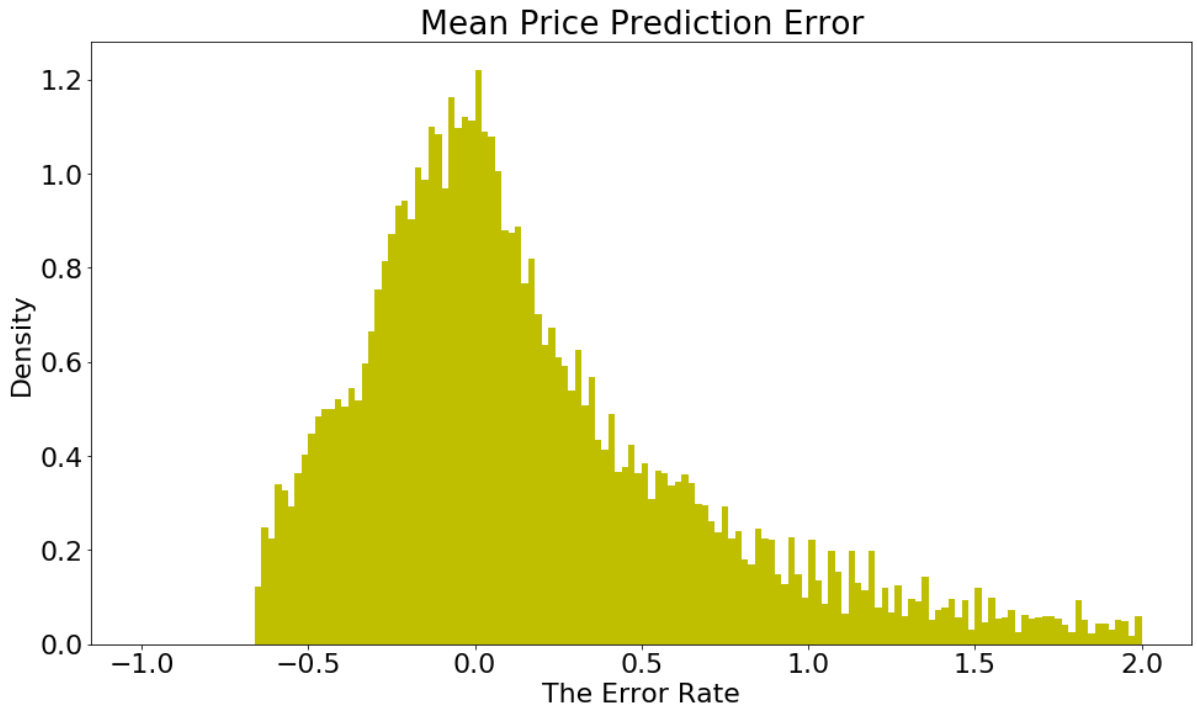
2.3.3 Make Prediction on Testing Set

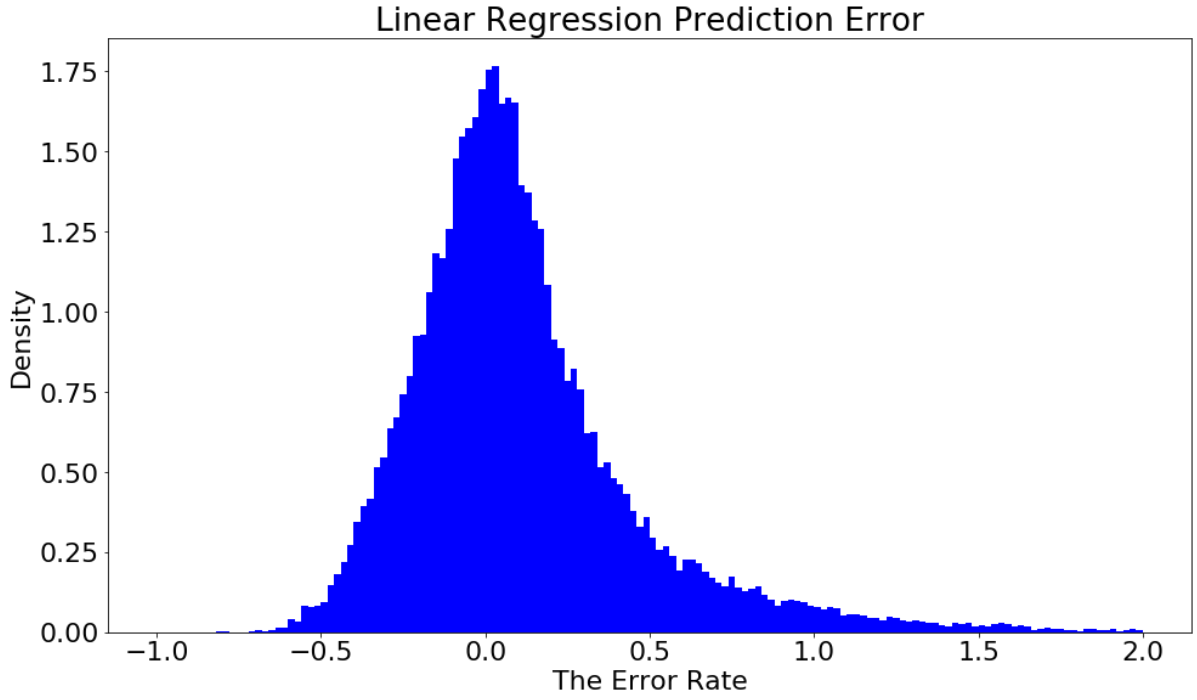
In [126]:
```
y_pred_rf = clf_rf.predict(X_test)
```

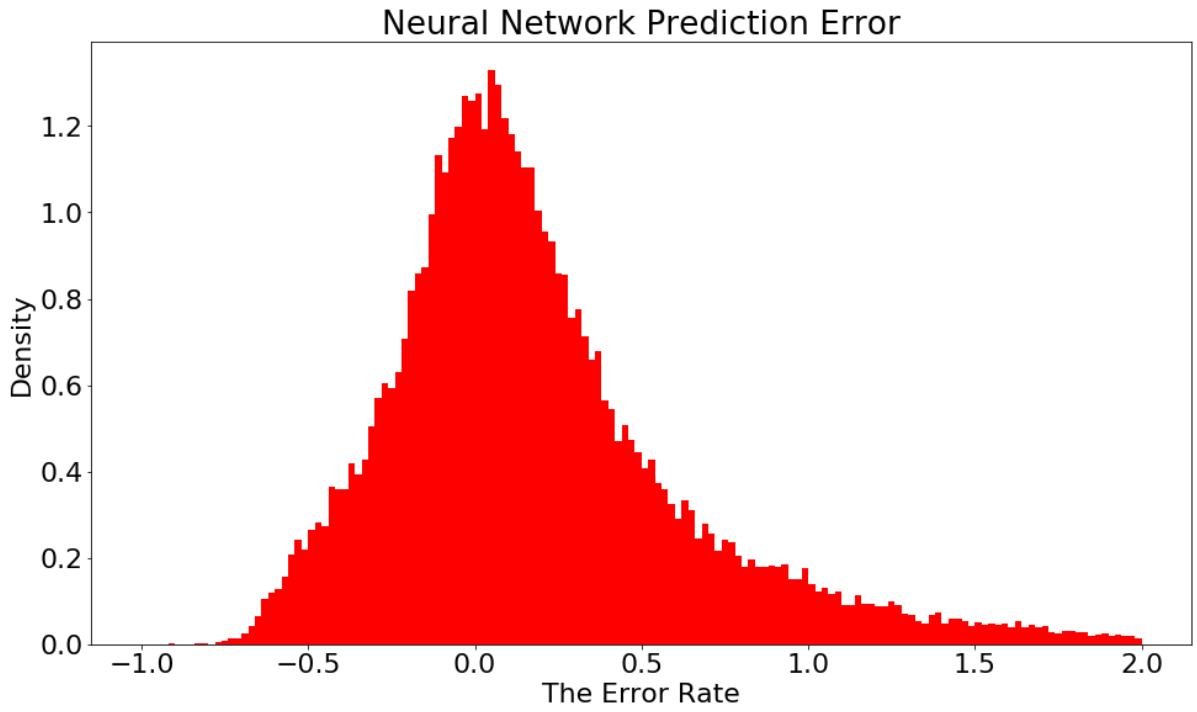
3 Evaluation

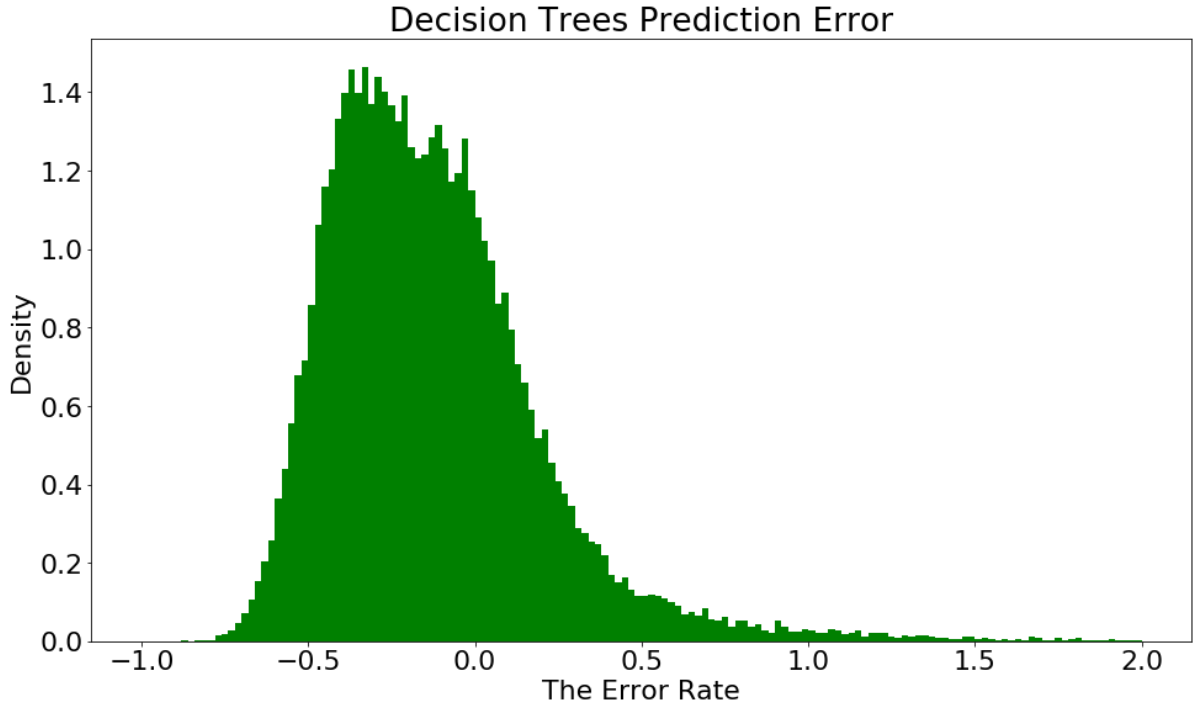
In [127]:
```
# Number of Algorithms (Prediction using the mean price is included)
ALGORITM_NUMBER = 5
# Name of Algorithms
ALGORITHM = ['Mean Price', 'Linear Regression', 'Neural Network', 'Decision Tr
ees', 'Random Forest']
# Color of Plot
COLOR_OF_PLOT = ['y','b','r', 'g','m']

# Calculate the differences between true values of prices and our predictions.
y_result = np.zeros((len(y_pred), 4 + ALGORITM_NUMBER))

# Calculate the mean price
sum = 0
for i in range(len(y_pred)):
    sum = sum + y_test[i]
y_mean = sum/len(y_pred)

# Calculate the results of predictions from different methods
for i in range(len(y_pred)):
    y_result[i][0] = y_test[i]
    y_result[i][1] = y_pred[i]
    y_result[i][2] = y_pred_nn[i]
    y_result[i][3] = (y_mean - y_test[i]) / y_test[i]
    y_result[i][4] = (y_pred[i] - y_test[i]) / y_test[i]
    y_result[i][5] = (y_pred_nn[i] - y_test[i]) / y_test[i]
    y_result[i][6] = (y_pred_dt[i] - y_test[i]) / y_test[i]
    y_result[i][7] = (y_pred_rf[i] - y_test[i]) / y_test[i]

y_result_df = pd.DataFrame(y_result)
```

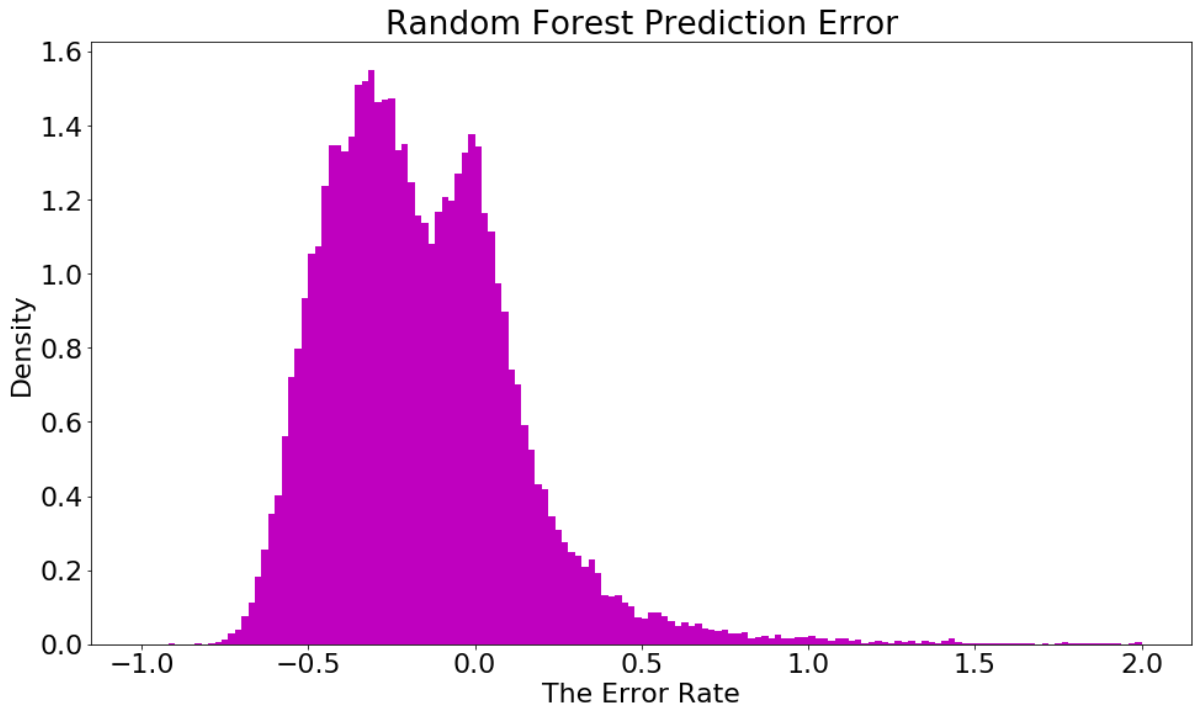
In [131]:
```
# Plot the distribution of error

for alg in range(ALGORITM_NUMBER):
    label_plt = ALGORITHM[alg]
    color_plt = COLOR_OF_PLOT[alg]
    plt.hist(y_result_df.iloc[:][alg + 3], bins=150, range = (-1,2), density =
True,
             label = label_plt, color = color_plt)
    plt.ylabel('Density')
    plt.xlabel('The Error Rate')
    plt.title(str(label_plt) + ' Prediction Error')
    plt.show()

for alg in range(ALGORITM_NUMBER):
    label_plt = ALGORITHM[alg]
    color_plt = COLOR_OF_PLOT[alg]
    if alg == 0:
        plt.hist(y_result_df.iloc[:][alg + 3], bins=150, range = (-1,2), densi
ty = True,
                 label = label_plt, color = color_plt)
    else:
        plt.hist(y_result_df.iloc[:][alg + 3], bins=150, range = (-1,2), densi
ty = True,
                 alpha=0.7, label = label_plt, color = color_plt)
plt.ylabel('Density')
plt.legend()
plt.title('Camparison of The Error Rate')
plt.show()
```

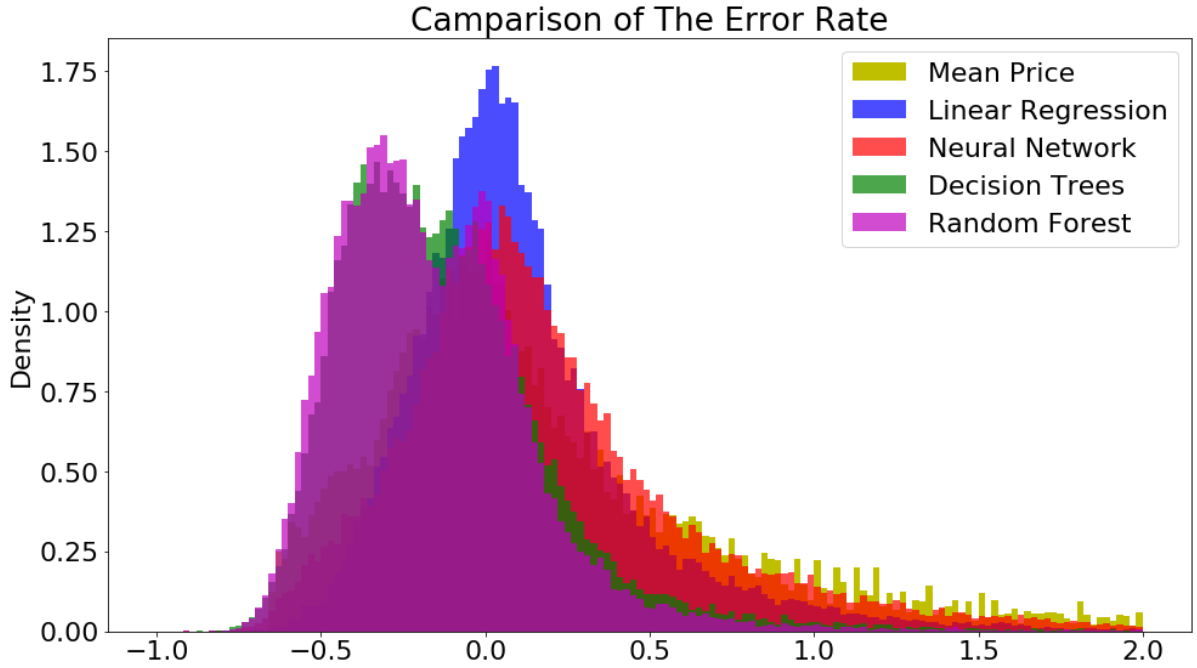

Mean Price Prediction Error
Density
The Error Rate
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Linear Regression Prediction Error
Density
The Error Rate
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Neural Network Prediction Error
Density
The Error Rate
0.0
0.2
0.4
0.6
0.8
1.0
1.2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

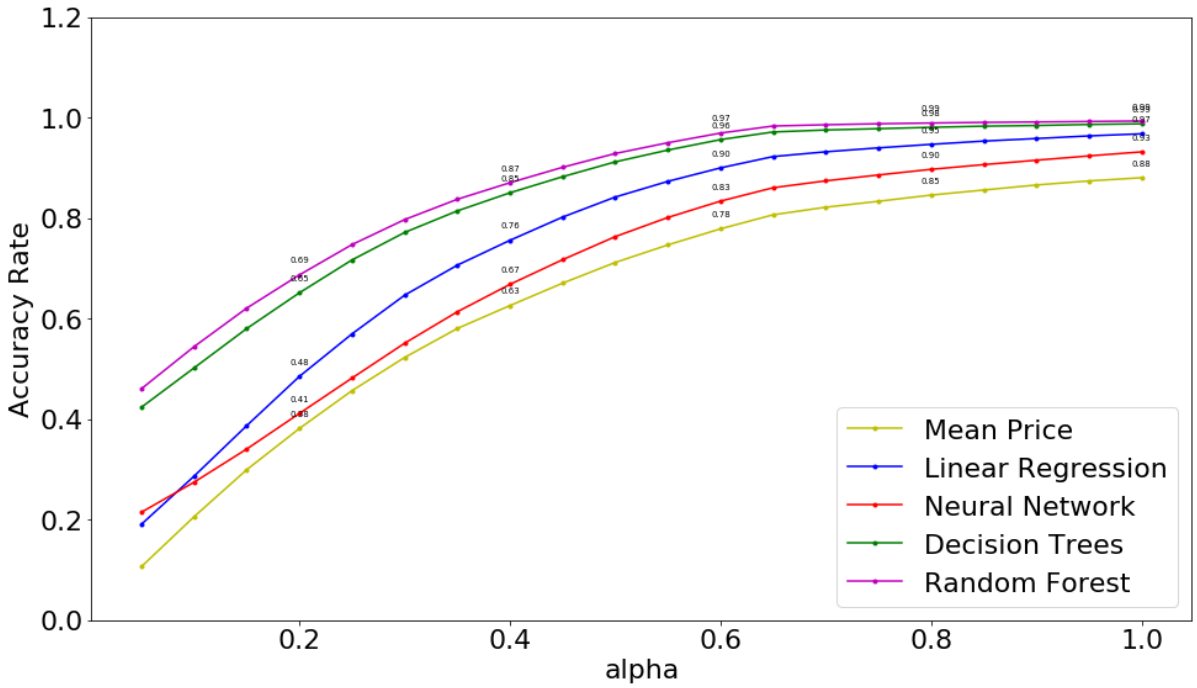

Decision Trees Prediction Error
Density
The Error Rate
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Random Forest Prediction Error
Density
The Error Rate
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Camparison of The Error Rate
Density
Mean Price
Linear Regression
Neural Network
Decision Trees
Random Forest
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

In [129]:
```
# Calcualte and plot the accuracy rate

critiria = []
accuracy = []*(ALGORITM_NUMBER)
for i in range(1,21):
    critiria.append(i*0.05)


for alg in range(ALGORITM_NUMBER):
    accuracy_temp = []
    for cri in critiria:
        correct_count = 0
        for i in range(len(y_result_df.index)):
            if y_result[i][alg+3] < cri and y_result[i][3] > -cri:
                correct_count = correct_count + 1
        accuracy_temp.append(correct_count/len(y_result_df.index))
    accuracy.append(accuracy_temp)
```

In [132]:
```
for alg in range(ALGORITM_NUMBER):
    color_plt = COLOR_OF_PLOT[alg]
    label_plt = ALGORITHM[alg]
    plt.plot(critiria,accuracy[alg][:],color_plt,label=label_plt, marker='.')

plt.ylim(top=1.2)
plt.ylim(bottom=0)


for alg in range(ALGORITM_NUMBER):
    counter = 0
    for x,y in zip(critiria,accuracy[alg][:]):
        label = "{:.2f}".format(y)
        if counter % 4 == 3:
            plt.annotate(label, (x,y), textcoords="offset points", xytext=(0,1
0), ha='center', fontsize=7)
        counter = counter + 1

plt.legend()
plt.xlabel('alpha')
plt.ylabel('Accuracy Rate')
plt.show()
```



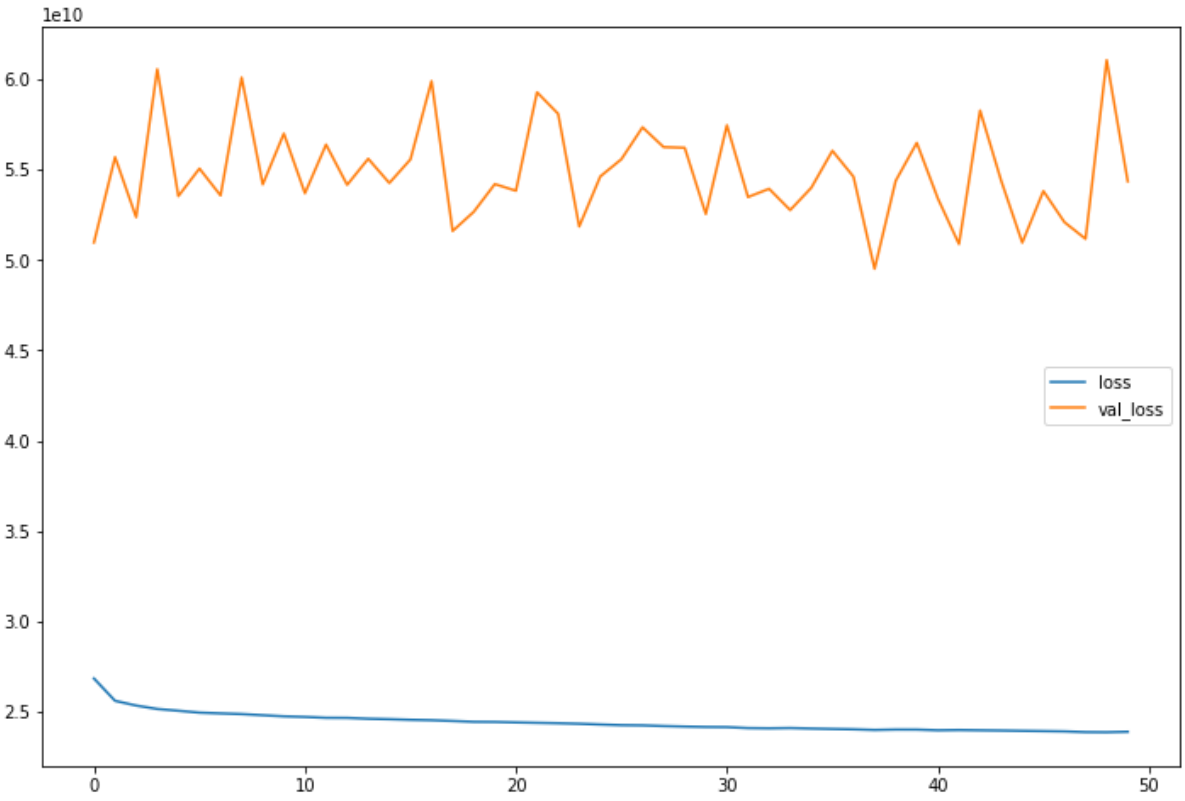
In [45]:
```
ALGORITM_NUMBER = 2

critiria = []
accuracy = []*(ALGORITM_NUMBER+1)
```

In [49]:
```
accuracy[0][:]
```

Out[49]:
```
[0.10677840638714112,
 0.2060223126949717,
 0.2995294242055729,
 0.38143076190979747,
 0.4567228890181357,
 0.5227356844498493,
 0.580368000845979,
 0.6259186802728283,
 0.6704647596891027,
 0.711653360122667,
 0.7465764289113308,
 0.7785650081954212,
 0.8064823137524454,
 0.8215513139110665,
 0.8333421456141279,
 0.8456881510072437,
 0.8557341511129911,
 0.8657537143763548,
 0.8738433881457198,
 0.8801882303177708]
```

In [27]:
```
loss_df = pd.DataFrame(model.history.history)
loss_df.plot(figsize=(12,8))
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2228b2984c8>
```



In [36]:
```
y_result
```

Out[36]:
```
array([[ 5.36117720e+05,  7.46004646e+05,  2.09886926e+05,
         3.91494103e-01],
       [ 6.52555790e+05,  6.43142427e+05, -9.41336297e+03,
        -1.44253765e-02],
       [ 9.20142140e+05,  8.20128927e+05, -1.00013213e+05,
        -1.08693221e-01],
       ...,
       [ 6.04949930e+05,  5.79699337e+05, -2.52505930e+04,
        -4.17399718e-02],
       [ 5.08836390e+05,  6.06320886e+05,  9.74844964e+04,
         1.91583185e-01],
       [ 2.24229540e+05,  5.22371962e+05,  2.98142422e+05,
         1.32963044e+00]])
```

In [ ]:
```

```