**Experiment:**

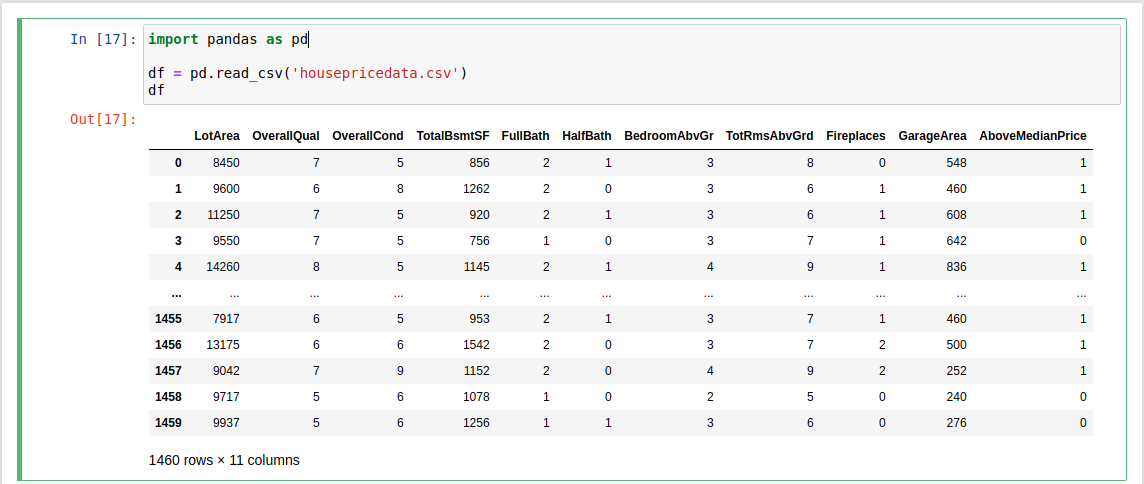
1. Create a neural network for the given ‘houseprice.csv’ to predict the whether price of the house is above or below median value or not

**Code :**

import pandas as pd

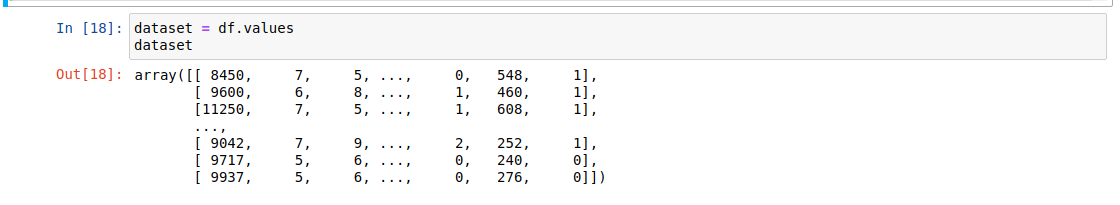
df = pd.read\_csv('housepricedata.csv')

df



dataset = df.values

dataset

X = dataset[:,0:10]

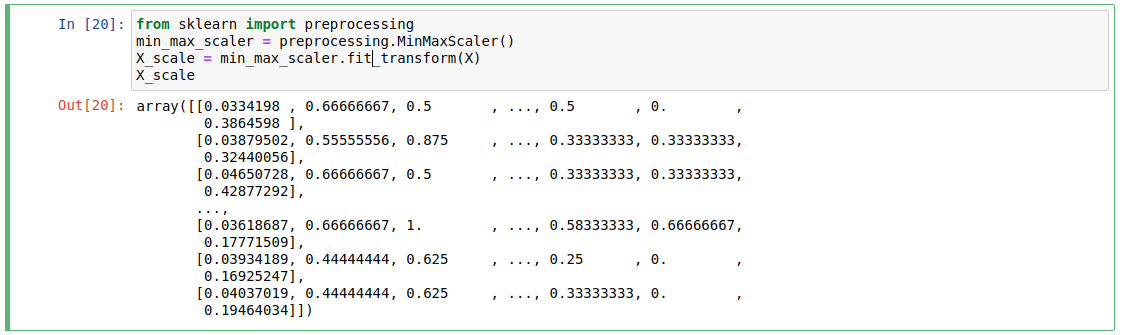
Y = dataset[:,10]

from sklearn import preprocessing

min\_max\_scaler = preprocessing.MinMaxScaler()

X\_scale = min\_max\_scaler.fit\_transform(X)

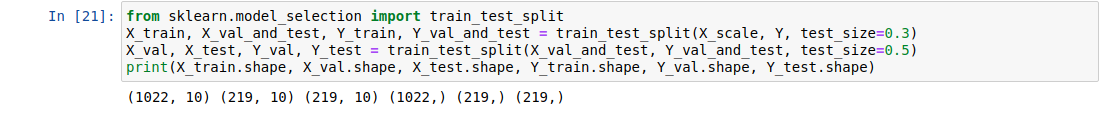
X\_scale

from sklearn.model\_selection import train\_test\_split

X\_train, X\_val\_and\_test, Y\_train, Y\_val\_and\_test = train\_test\_split(X\_scale, Y, test\_size=0.3)

X\_val, X\_test, Y\_val, Y\_test = train\_test\_split(X\_val\_and\_test, Y\_val\_and\_test, test\_size=0.5)

print(X\_train.shape, X\_val.shape, X\_test.shape, Y\_train.shape, Y\_val.shape, Y\_test.shape)

from keras.models import Sequential

from keras.layers import Dense

model = Sequential([

Dense(32, activation='relu', input\_shape=(10,)),

Dense(32, activation='relu'),

Dense(1, activation='sigmoid'),

])

model.compile(optimizer='sgd',

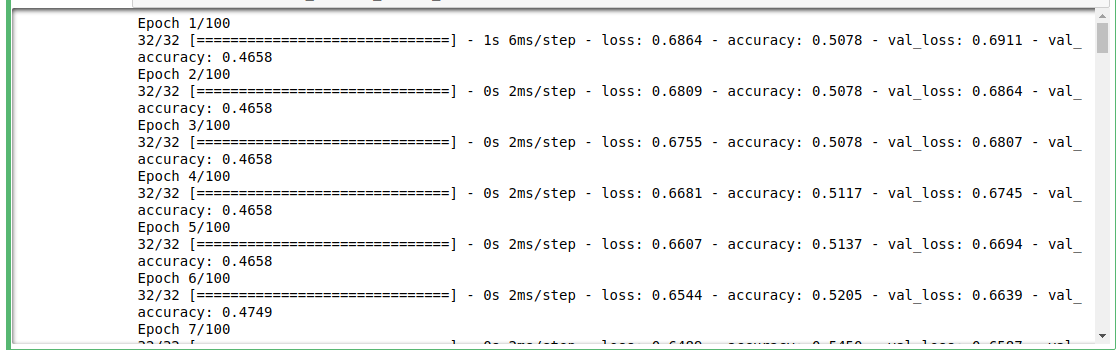
loss='binary\_crossentropy',

metrics=['accuracy'])

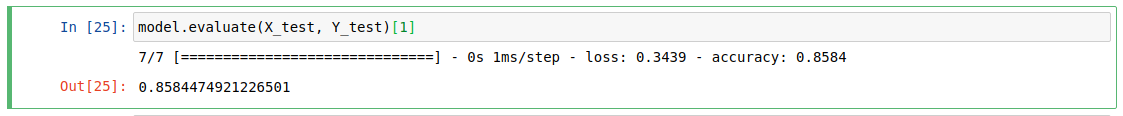
hist = model.fit(X\_train, Y\_train,

batch\_size=32, epochs=100,

validation\_data=(X\_val, Y\_val))



model.evaluate(X\_test, Y\_test)[1]

import matplotlib.pyplot as plt

plt.plot(hist.history['loss'])

plt.plot(hist.history['val\_loss'])

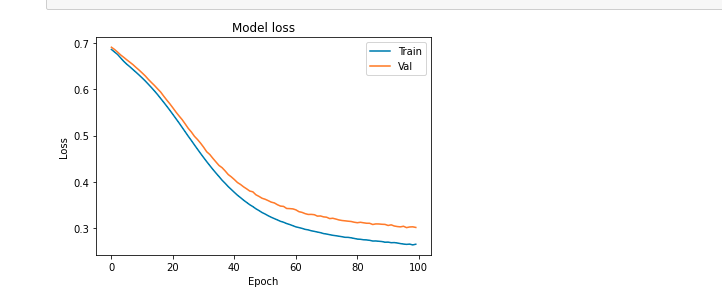
plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Val'], loc='upper right')

plt.show()



plt.plot(hist.history['accuracy'])

plt.plot(hist.history['val\_accuracy'])

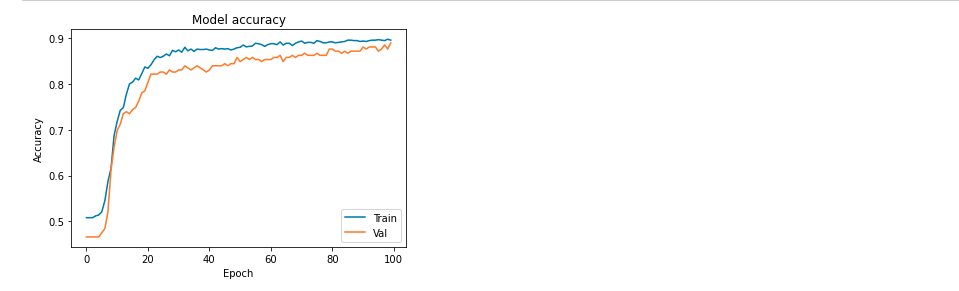
plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Val'], loc='lower right')

plt.show()



model\_2 = Sequential([

Dense(1000, activation='relu', input\_shape=(10,)),

Dense(1000, activation='relu'),

Dense(1000, activation='relu'),

Dense(1000, activation='relu'),

Dense(1, activation='sigmoid'),

])

model\_2.compile(optimizer='adam',

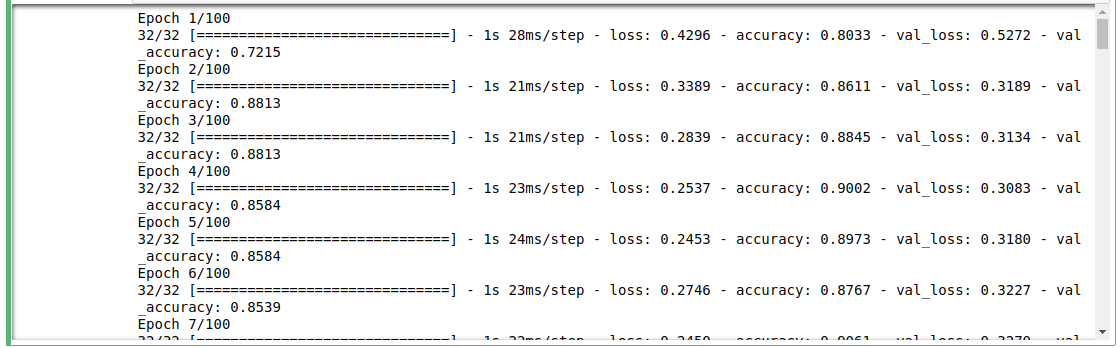
loss='binary\_crossentropy',

metrics=['accuracy'])

hist\_2 = model\_2.fit(X\_train, Y\_train,

batch\_size=32, epochs=100,

validation\_data=(X\_val, Y\_val))



plt.plot(hist\_2.history['loss'])

plt.plot(hist\_2.history['val\_loss'])

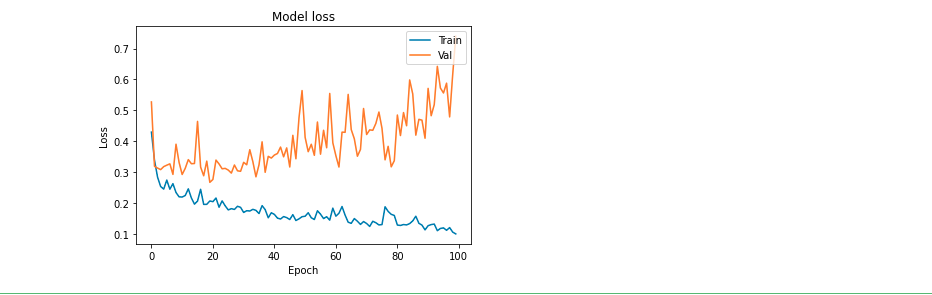
plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Val'], loc='upper right')

plt.show()



plt.plot(hist\_2.history['accuracy'])

plt.plot(hist\_2.history['val\_accuracy'])

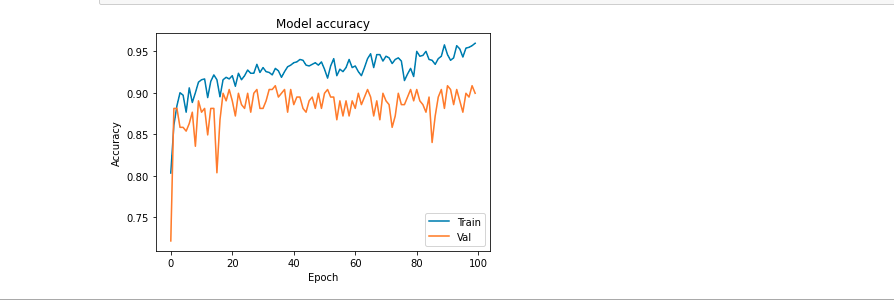
plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Val'], loc='lower right')

plt.show()



from keras.layers import Dropout

from keras import regularizers

model\_3 = Sequential([

Dense(1000, activation='relu', kernel\_regularizer=regularizers.l2(0.01), input\_shape=(10,)),

Dropout(0.3),

Dense(1000, activation='relu', kernel\_regularizer=regularizers.l2(0.01)),

Dropout(0.3),

Dense(1000, activation='relu', kernel\_regularizer=regularizers.l2(0.01)),

Dropout(0.3),

Dense(1000, activation='relu', kernel\_regularizer=regularizers.l2(0.01)),

Dropout(0.3),

Dense(1, activation='sigmoid', kernel\_regularizer=regularizers.l2(0.01)),

])

model\_3.compile(optimizer='adam',

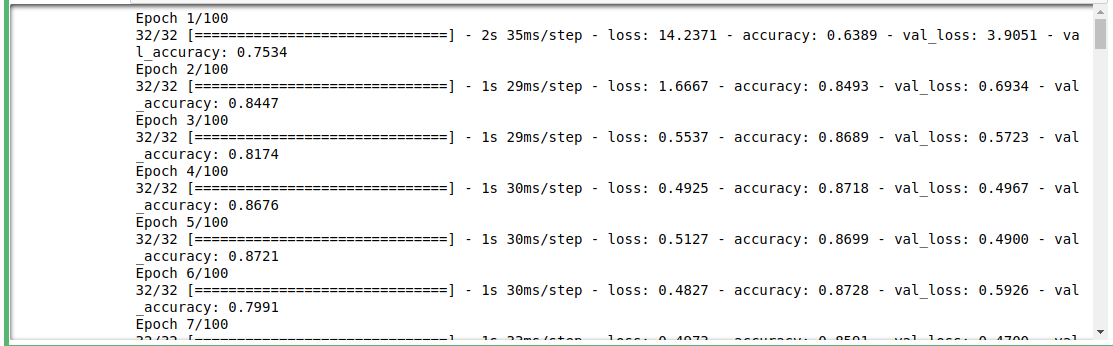
loss='binary\_crossentropy',

metrics=['accuracy'])

hist\_3 = model\_3.fit(X\_train, Y\_train,

batch\_size=32, epochs=100,

validation\_data=(X\_val, Y\_val))



plt.plot(hist\_3.history['loss'])

plt.plot(hist\_3.history['val\_loss'])

plt.title('Model loss')

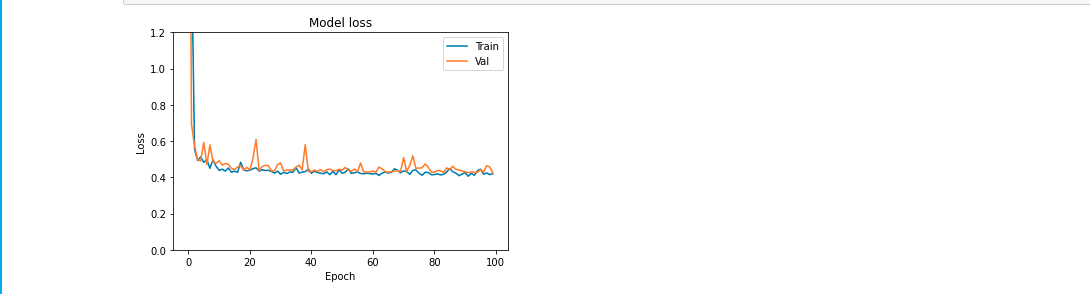
plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Val'], loc='upper right')

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

plt.show()



plt.plot(hist\_3.history['accuracy'])

plt.plot(hist\_3.history['val\_accuracy'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Val'], loc='lower right')

plt.show()

