```
pip install psycopg2-binary
```

```
    Collecting psycopg2-binary
      Downloading https://files.pythonhosted.org/packages/6d/45/c519a5cfac05e14b1ccb
         |████████████████████████████████| 3.0MB 13.9MB/s
    Installing collected packages: psycopg2-binary
    Successfully installed psycopg2-binary-2.8.6
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,OneHotEncoder
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
import tensorflow as tf
import sqlalchemy as db
from sqlalchemy.ext.automap import automap_base
from sklearn.metrics import mean_squared_error
from keras.metrics import RootMeanSquaredError
from datetime import date
```

```
engine = db.create_engine('postgresql://postgres:finalproject@finalproject.cfnecwioim
with engine.connect() as conn, conn.begin():
    df = pd.read_sql("""select * from all_cleaned_redfin_data;""", conn)
```

```
ml_df = df
ml_df = ml_df.drop([
    'address',
    'state_or_province',
    #'city',
    'url',
    'mls_number',
    'latitude',
    'longitude',
    'neighborhood',
    'price_per_square_feet'
    ], axis =1)
ml_df
```

	index	property_type	city	zip_code	price	beds	baths	s
0	0	Single Family Residential	Nashville	37215	1425000	4.0	5.0	
1	1	Single Family Residential	Nashville	37220	1975000	5.0	4.5	
2	2	Single Family Residential	Nashville	37215	1100000	4.0	5.0	
3	3	Single Family Residential	Nashville	37215	1600000	4.0	4.0	
4	4	Single Family Residential	Nashville	37215	1775000	4.0	5.0	
...	...	...	...	...	...	...	...	...
5918	340	Single Family Residential	Madison	37115	235950	1.0	1.0	
5919	341	Single Family Residential	Madison	37115	230950	1.0	1.0	
5920	342	Single Family Residential	Madison	37115	200950	1.0	1.0	
5921	343	Single Family Residential	Madison	37115	225950	1.0	1.0	

age = today - year built zip code price per square feet try scaling everything loss function or roc curve

Commented out year_built as string calculated age and droped year built

```
convert_dict = {'zip_code': object,
                'beds': int,
                'square_feet': int,
                'lot_size': int,
                #'year_built': object,
               }
ml_df['age_in_years'] = date.today().year - ml_df['year_built']
ml_df = ml_df.drop(['year_built'], axis=1)
ml_df = ml_df.astype(convert_dict)
object1 = ml_df.dtypes[ml_df.dtypes == "object"].index.to_list()
object1.remove('status')
print(object1)
```

```
['property_type', 'city', 'zip_code']
```

```
enc = OneHotEncoder(sparse=False)
```

```
encode df = pd.DataFrame(enc.fit transform(ml df[object1]))
```

```
encode_df.columns = enc.get_feature_names(object1)
encode_df

ml_encode_df = ml_df.merge(encode_df,left_index=True, right_index=True)
ml_encode_df = ml_encode_df.drop(object1,1)
ml_encode_df

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

# Scaling the Price column of the created dataFrame and storing
# the result in ScaledPrice Column
ml_encode_df[["scaledSF", "scaledLS", "scaledAge"]] = scaler.fit_transform(ml_encode_
ml_encode_df = ml_encode_df.drop(["square_feet","lot_size", "age_in_years"], axis= 1)
ml_encode_df
```

	index	price	beds	baths	status	property_type_Condo/Co-op	p
0	0	1425000	4	5.0	Sold	0.0	
1	1	1975000	5	4.5	Sold	0.0	
2	2	1100000	4	5.0	Sold	0.0	
3	3	1600000	4	4.0	Sold	0.0	
4	4	1775000	4	5.0	Sold	0.0	
...	...	...	...	...	...	...	
5918	340	235950	1	1.0	Active	0.0	
5919	341	230950	1	1.0	Active	0.0	
5920	342	200950	1	1.0	Active	0.0	
5921	343	225950	1	1.0	Active	0.0	
5922	344	195950	1	1.0	Active	0.0	

```
active_df = ml_encode_df.loc[ml_encode_df['status'] == 'Active'].drop(['status'], axi
sold_df = ml_encode_df.loc[ml_encode_df['status'] != 'Active'].drop(['status'], axis=
y_sold, y_active = sold_df["price"].values, active_df["price"].values
X_sold, X_active = sold_df.drop(["price"],1).values, active_df.drop(["price"],1).valu


X_train, X_test, y_train, y_test = train_test_split(X_sold, y_sold, random_state=78)


#Define the model - deep neural net
number_input_features = len(X_train[0])
```

```
number_input_features = len(X_train[0])

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=50, input_dim=number_input_features, activation="relu
)
nn.add(
    tf.keras.layers.Dense(units=40, activation="relu")
)

nn.add(
    tf.keras.layers.Dense(units=30, activation="relu")
)
nn.add(
    tf.keras.layers.Dense(units=20, activation="relu")
)

nn.add(
    tf.keras.layers.Dense(units=10, activation="relu")
)
# Second hidden layer
nn.add(tf.keras.layers.Dense(units=8, kernel_initializer='normal'))



# Output layer
nn.add(tf.keras.layers.Dense(units=1))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 50)                3200
_________________________________________________________________
dense_15 (Dense)             (None, 40)                2040
_________________________________________________________________
dense_16 (Dense)             (None, 30)                1230
_________________________________________________________________
dense_17 (Dense)             (None, 20)                620
_________________________________________________________________
dense_18 (Dense)             (None, 10)                210
_________________________________________________________________
dense_19 (Dense)             (None, 8)                 88
_________________________________________________________________
dense_20 (Dense)             (None, 1)                 9
=================================================================
Total params: 7,397
Trainable params: 7,397
```

```
     Non-trainable params: 0

# def baseline_model():
#   # create model
#   model = Sequential()
#   model.add(Dense(13, input_dim= len(X_train[0]), kernel_initializer='he_normal', a
#   model.add(Dense(1, kernel_initializer='normal'))
#   # Compile model
#   model.compile(loss='mean_squared_error', optimizer='Adam')
#   return model
# #evaluate the model
# param_grid = {
# #     'optimizer': ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nad
#      'batch_size': [10, 20, 40, 60, 80, 100],
#      'epochs': [100, 200, 300],
# #     'learn_rate': [0.001, 0.01, 0.1, 0.2, 0.3],
# #     'momentum': [0.0, 0.2, 0.4, 0.6, 0.8, 0.9],
# #     'init_mode': ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal',
# #     'activation': ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid',
# #     'weight_constraint': [1, 2, 3, 4, 5],
# #     'dropout_rate': [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9],
# #     'neurons': [1, 5, 10, 15, 20, 25, 30]

# }

# model = KerasRegressor(build_fn=baseline_model, epochs=100, batch_size=5, verbose=0
# grid_search = GridSearchCV(estimator = model, param_grid = param_grid,
#                       cv = 3, n_jobs = -1, verbose = 2, return_train_score=True)
# grid_result = grid_search.fit(X_train, y_train)
# print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
# means = grid_result.cv_results_['mean_test_score']
# stds = grid_result.cv_results_['std_test_score']
# params = grid_result.cv_results_['params']
# for mean, stdev, param in zip(means, stds, params):
#     print("%f (%f) with: %r" % (mean, stdev, param))



# Compile the model
nn.compile(loss='mse', optimizer="adam", metrics=[tf.keras.metrics.RootMeanSquaredErr


# Train the model
fit_model = nn.fit(X_train,y_train,epochs=200)


prediction = nn.predict(X_test)
pred = pd.DataFrame({ 'actual': y_test})
pred['prediction'] = prediction
pred['error'] =pred.prediction - pred.actual
pred['error_abs'] = abs(pred['error'])
print(pred.error_abs.mean())
print(mean_squared_error(y_test, prediction))
```

```
print(mean_squared_error(y_test, prediction))
print(mean_squared_error(y_test, prediction, squared=False))
```

```
93391.0855540985
32997114415.376217
181651.07876193913
```

```
# Train the model
fit_model = nn.fit(X_train,y_train,epochs=100)
```

```
prediction = nn.predict(X_test)
pred = pd.DataFrame({ 'actual': y_test})
pred['prediction'] = prediction
pred['error'] =pred.prediction - pred.actual
pred['error_abs'] = abs(pred['error'])
print(pred.error_abs.mean())
print(mean_squared_error(y_test, prediction))
print(mean_squared_error(y_test, prediction, squared=False))
```

```
87604.24701057965
30339684543.100517
174182.90542731373
```

```
# Train the model
fit_model = nn.fit(X_train,y_train,epochs=100)
```

```
prediction = nn.predict(X_test)
pred = pd.DataFrame({ 'actual': y_test})
pred['prediction'] = prediction
pred['error'] =pred.prediction - pred.actual
pred['error_abs'] = abs(pred['error'])
print(pred.error_abs.mean())
print(mean_squared_error(y_test, prediction))
print(mean_squared_error(y_test, prediction, squared=False))
```

```
92964.18187122155
31790344500.400116
178298.47026937758
```

```
# Train the model
fit_model = nn.fit(X_train,y_train,epochs=100)
```

```
prediction = nn.predict(X_test)
pred = pd.DataFrame({ 'actual': y_test})
pred['prediction'] = prediction
pred['error'] =pred.prediction - pred.actual
pred['error_abs'] = abs(pred['error'])
print(pred.error_abs.mean())
print(mean_squared_error(y_test, prediction))
```

```
print(mean_squared_error(y_test, prediction, squared=False))
```

```
90667.0099128734
35046097701.83956
187206.03008941663
```

```
# Train the model
fit_model = nn.fit(X_train,y_train,epochs=100)
```

```
prediction = nn.predict(X_test)
pred = pd.DataFrame({ 'actual': y_test})
pred['prediction'] = prediction
pred['error'] =pred.prediction - pred.actual
pred['error_abs'] = abs(pred['error'])
print(pred.error_abs.mean())
print(mean_squared_error(y_test, prediction))
print(mean_squared_error(y_test, prediction, squared=False))
```

```
118584.6579669719
56754310369.16693
238231.6317560851
```

```
 #Train the model
fit_model = nn.fit(X_train,y_train,epochs=100)
```

```
prediction = nn.predict(X_test)
pred = pd.DataFrame({ 'actual': y_test})
pred['prediction'] = prediction
pred['error'] =pred.prediction - pred.actual
pred['error_abs'] = abs(pred['error'])
print(pred.error_abs.mean())
print(mean_squared_error(y_test, prediction))
print(mean_squared_error(y_test, prediction, squared=False))
```

```
93744.06381690079
44074800473.11699
209939.9925529126
```

```
 #Train the model
fit_model = nn.fit(X_train,y_train,epochs=100)
```

```
prediction = nn.predict(X_test)
pred = pd.DataFrame({ 'actual': y_test})
pred['prediction'] = prediction
pred['error'] =pred.prediction - pred.actual
pred['error_abs'] = abs(pred['error'])
print(pred.error_abs.mean())
print(mean_squared_error(y_test, prediction))
print(mean squared error(y test, prediction, squared=False))
```

```
103831.73846461593
48460593813.927055
220137.67013831835
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in
_input_request(self, prompt, ident, parent, password)
    728             try:
--> 729                 ident, reply = self.session.recv(self.stdin_socket, 0)
    730             except Exception:

                                6 frames

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket._recv_copy()

KeyboardInterrupt:

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in
_input_request(self, prompt, ident, parent, password)
    732             except KeyboardInterrupt:
    733                 # re-raise KeyboardInterrupt, to truncate traceback
--> 734                 raise KeyboardInterrupt
    735             else:
    736                 break

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

```
import numpy as np
active_df = df.loc[df['status'] == 'Active'].drop(['status'], axis=1)
active_df['predicted_price'] = nn.predict(X_active)
active_df['house_value'] = np.where(active_df['price'] > active_df['predicted_price']
active_df.groupby(['house_value']).count()['mls_number']
active_df.to_json('house_value.json')
!cp house_value.json "drive/My Drive/"
```

```
active_df
```

	property_type	address	city	state_or_province	zip_code	price	beds
0	Single Family Residential	413 Brook View Estates Dr	Antioch	TN	37013	244900	3.0
1	Single Family Residential	135 Spring Valley Rd	Nashville	TN	37214	340000	2.0
2	Single Family Residential	2509 Slaydon Dr	Nashville	TN	37207	325000	3.0
5	Condo/Co-op	121 Beech Forge Dr	Antioch	TN	37013	200000	2.0
7	Single Family Residential	1632 Aaronwood Dr	Old Hickory	TN	37138	255900	3.0
...	...	...	...	...	...	...	...
340	Single Family Residential	624 W Due West Ave #401	Madison	TN	37115	235950	1.0
341	Single Family Residential	624 W Due West Ave #301	Madison	TN	37115	230950	1.0
342	Single Family Residential	624 W Due West Ave #205	Madison	TN	37115	200950	1.0
343	Single Family Residential	624 W Due West Ave #201	Madison	TN	37115	225950	1.0
344	Single Family Residential	624 W Due West Ave #105	Madison	TN	37115	195950	1.0

4s completed at 8:33 PM