Part B Question 2

In [ ]:
```
# Setting the seed here is sufficient.
# If you don't plan to use these starter code, make sure you add this cell.

SEED = 42

import os
os.environ['TF_CUDNN_DETERMINISTIC'] = '1'
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
import random
random.seed(SEED)

import numpy as np
np.random.seed(SEED)

import tensorflow as tf
tf.random.set_seed(SEED)
```

In [ ]:
```
import graphviz
import pydot_ng as pydot
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Normalization, StringLookup, IntegerLookup
from math import floor
from math import sqrt
```

In [ ]:
```
import pandas as pd
df = pd.read_csv('hdb_price_prediction.csv')
df
```

Out[ ]:

	month	year	full_address	nearest_stn	dist_to_nearest_stn	dist_to_dhoby	degree_centrality	e
0	1	2017	406 ANG MO KIO AVENUE 10	Ang Mo Kio	1.007264	7.006044	0.016807	
1	1	2017	108 ANG MO KIO AVENUE 4	Ang Mo Kio	1.271389	7.983837	0.016807	
2	1	2017	602 ANG MO KIO AVENUE 5	Yio Chu Kang	1.069743	9.090700	0.016807	
3	1	2017	465 ANG MO KIO AVENUE 10	Ang Mo Kio	0.946890	7.519889	0.016807	
4	1	2017	601 ANG MO KIO AVENUE 5	Yio Chu Kang	1.092551	9.130489	0.016807	
...	...	...	...	...	...	...	...	
133407	6	2022	877 YISHUN STREET 81	Khatib	0.475885	12.738721	0.016807	
133408	1	2022	633 YISHUN STREET 61	Khatib	0.774113	13.229106	0.016807	
133409	2	2022	633 YISHUN STREET 61	Khatib	0.774113	13.229106	0.016807	
133410	2	2022	632 YISHUN STREET 61	Khatib	0.700595	13.222912	0.016807	
133411	5	2022	605 YISHUN STREET 61	Khatib	0.603845	13.592586	0.016807	

133412 rows × 13 columns

In [ ]:

```
# The functions in this cell are adapted from https://keras.io/examples/structured_data/
# It is the same link as the one mentioned in the question paper (Q1b)

def dataframe_to_dataset(dataframe):
    dataframe = dataframe.copy()

    labels = dataframe.pop("resale_price")
    ds = tf.data.Dataset.from_tensor_slices((dict(dataframe), labels))
    ds = ds.shuffle(buffer_size=len(dataframe))
    return ds


def encode_numerical_feature(feature, name, dataset):
    # Create a Normalization layer for our feature
    normalizer = Normalization()

    # Prepare a Dataset that only yields our feature
```

```
    feature_ds = dataset.map(lambda x, y: x[name])
    feature_ds = feature_ds.map(lambda x: tf.expand_dims(x, -1))

    # Learn the statistics of the data
    normalizer.adapt(feature_ds)

    # Normalize the input feature
    encoded_feature = normalizer(feature)
    return encoded_feature


def encode_categorical_feature(feature, name, dataset, is_string):
    lookup_class = StringLookup if is_string else IntegerLookup
    # Create a lookup layer which will turn strings into integer indices
    lookup = lookup_class(output_mode="binary") # NOTE: as mentioned in the question pa

    # Prepare a Dataset that only yields our feature
    feature_ds = dataset.map(lambda x, y: x[name])
    feature_ds = feature_ds.map(lambda x: tf.expand_dims(x, -1))

    # Learn the set of possible string values and assign them a fixed integer index
    lookup.adapt(feature_ds)

    # Turn the string input into integer indices
    encoded_feature = lookup(feature)
    return encoded_feature
```

In [ ]:
```
from keras import backend as K
def r2(y_true, y_pred):
    '''
    # Obtained from https://jmlb.github.io/ml/2017/03/20/CoeffDetermination_CustomMetri
    # TODO: you have to find out how to use it in your code
    '''
    SS_res = K.sum(K.square( y_true - y_pred ))
    SS_tot = K.sum(K.square( y_true - K.mean(y_true) ) )
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )
```

From Question 1

In [ ]:
```
# Split data

train_dataframe = df[df['year']<= 2020]
test_dataframe =  df[df['year']>2020]
```

In [ ]:
```
category_not_used = ["full_address", "nearest_stn"]
train_dataframe = train_dataframe.drop(category_not_used, axis = 1)
test_dataframe = test_dataframe.drop(category_not_used, axis = 1)

train_ds = dataframe_to_dataset(train_dataframe)
test_ds = dataframe_to_dataset(test_dataframe)

train_ds = train_ds.batch(256)
test_ds = test_ds.batch(256)
```

In [ ]:
```
#Categorical feature encoded as integer
month = keras.Input(shape=(1,), name="month", dtype="int64")
```

```
# Categorical feature encoded as string
flat_model_type = keras.Input(shape=(1,), name="flat_model_type", dtype="string")
storey_range = keras.Input(shape=(1,), name="storey_range", dtype="string")

# Numerical features
dist_to_nearest_stn = keras.Input(shape=(1,), name="dist_to_nearest_stn")
dist_to_dhoby = keras.Input(shape=(1,), name="dist_to_dhoby")
degree_centrality = keras.Input(shape=(1,), name="degree_centrality")
eigenvector_centrality = keras.Input(shape=(1,), name="eigenvector_centrality")
remaining_lease_years = keras.Input(shape=(1,), name="remaining_lease_years")
floor_area_sqm = keras.Input(shape=(1,), name="floor_area_sqm")
```

In [ ]:
```
all_inputs = [month,flat_model_type,storey_range,dist_to_nearest_stn,
            dist_to_dhoby,degree_centrality,eigenvector_centrality,remaining_lease_year
```

Question 2

Question 2A

Further split of training dataset to train and validation

In [ ]:
```
Q2_validation_dataframe = train_dataframe[train_dataframe["year"]==2020]
Q2_train_dataframe = train_dataframe[train_dataframe["year"]<2020]
```

In [ ]:
```
Q2_train_ds = dataframe_to_dataset(Q2_train_dataframe)
Q2_val_ds = dataframe_to_dataset(Q2_validation_dataframe)

Q2_train_ds = Q2_train_ds.batch(256)
Q2_val_ds = Q2_val_ds.batch(256)
```

Question 2B

In [ ]:
```
def Q2_encode_categorical_feature(feature, name, dataset, is_string, num_categories, di
    lookup_class = StringLookup if is_string else IntegerLookup
    # Create a lookup layer which will turn strings into integer indices
    lookup = lookup_class(output_mode="int")

    # Prepare a Dataset that only yields our feature
    feature_ds = dataset.map(lambda x, y: x[name])
    feature_ds = feature_ds.map(lambda x: tf.expand_dims(x,-1))

    # Learn the set of possible string values and assign them a fixed integer index
    lookup.adapt(feature_ds)

    # Turn the string input into integer indices
    encoded_feature = lookup(feature)

    emb = layers.Embedding(input_dim=num_categories+1, output_dim=floor(num_categories/
    embedded = emb(encoded_feature)

    return layers.Flatten()(embedded)
```

Question 2C

In [ ]:
```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
```

In [ ]:
```
# Numerical features encoded wit Q2_train_ds
dist_to_nearest_stn_encoded = encode_numerical_feature(dist_to_nearest_stn, "dist_to_nea
dist_to_dhoby_encoded = encode_numerical_feature(dist_to_dhoby, "dist_to_dhoby", Q2_tra
degree_centrality_encoded = encode_numerical_feature(degree_centrality, "degree_central
eigenvector_centrality_encoded = encode_numerical_feature(eigenvector_centrality, "eigen
remaining_lease_year_encoded = encode_numerical_feature(remaining_lease_years, "remainin
floor_area_sqm_encoded = encode_numerical_feature(floor_area_sqm,"floor_area_sqm", Q2_tr
```

In [ ]:
```
import keras_tuner
def build_model(hp):
    Q2_optimizer = tf.keras.optimizers.Adam(hp.Float('learning_rate', 1e-4, 2e-1, sampl
    divisor = hp.Int("divisor", min_value=1, max_value=2, step=1)

    hidden_units = hp.Int("hidden_units", min_value=4, max_value=32, step=4)


    month_num_categories = df["month"].nunique()
    flat_model_type_num_categories = df["flat_model_type"].nunique()
    storey_range_num_categories = df["storey_range"].nunique()


    #Integer categorical features
    month_embedded = Q2_encode_categorical_feature(month, "month",Q2_train_ds, False, mo
    #String categorical features
    flat_model_type_embedded = Q2_encode_categorical_feature(flat_model_type, "flat_mode
    storey_range_embedded = Q2_encode_categorical_feature(storey_range, "storey_range",Q

    Q2_all_features = layers.Concatenate()(
                    [
                        month_embedded,
                        storey_range_embedded,
                        flat_model_type_embedded,
                        floor_area_sqm_encoded,
                        remaining_lease_year_encoded,
                        degree_centrality_encoded,
                        eigenvector_centrality_encoded,
                        dist_to_nearest_stn_encoded,
                        dist_to_dhoby_encoded
                    ]
        )
    hidden_layer = layers.Dense(units=hidden_units, activation ="linear")(Q2_all_featur
    Q2_output = layers.Dense(1, activation="linear")(hidden_layer)
    Q2_model = keras.Model(all_inputs, Q2_output)
    Q2_model.compile(optimizer=Q2_optimizer, loss= "mse",metrics=[r2])

    return Q2_model
```

In [ ]:
```
tuner = keras_tuner.RandomSearch(
    build_model,
    objective='val_loss',
    max_trials=10)

tuner.search(Q2_train_ds, epochs=50, validation_data=Q2_val_ds, callbacks=[callback])
```

```python
best_model = tuner.get_best_models(1)[0]
best_hyperparameters = tuner.get_best_hyperparameters(1)[0]
```

```
Trial 10 Complete [00h 01m 21s]
val_loss: 206107623424.0

Best val_loss So Far: 3843214080.0
Total elapsed time: 00h 12m 29s
INFO:tensorflow:Oracle triggered exit
```

Best hyperparamters

In [ ]:
```python
best_hyperparameters.values
```

Out[ ]:
```
{'learning_rate': 0.046185127256095915, 'divisor': 2, 'hidden_units': 8}
```

Question 2D

Training of model based on best model configuration and test RMSE

In [ ]:
```python
Q2_best_model_history = {}
Q2_best_model = build_model(best_hyperparameters)

# Save best epoch only
callback_list = [
       tf.keras.callbacks.ModelCheckpoint(
            filepath='PartB_bestepoch/',
            save_freq='epoch', verbose=1, monitor='val_loss',
            save_weights_only=True, save_best_only=True
       )
]

# Train on the non-test split ( 2020 and before hence using train_ds instead of Q2_trai
Q2_best_model_history["best_model"] = Q2_best_model.fit(train_ds, epochs=50, validation_

# Save the model train with the optimal hyperparameters
Q2_best_model.save('PartB_best_model/')
```

```
Epoch 1/50
329/342 [===========================>..] - ETA: 0s - loss: 82699100160.0000 - r2: -2.52
23
Epoch 1: val_loss improved from inf to 17899706368.00000, saving model to PartB_bestepo
ch\
342/342 [==============================] - 3s 6ms/step - loss: 80105832448.0000 - r2: -
2.4058 - val_loss: 17899706368.0000 - val_r2: 0.3491
Epoch 2/50
342/342 [==============================] - ETA: 0s - loss: 9185911808.0000 - r2: 0.6118
Epoch 2: val_loss improved from 17899706368.00000 to 14939295744.00000, saving model to
PartB_bestepoch\
342/342 [==============================] - 2s 5ms/step - loss: 9185911808.0000 - r2: 0.
6118 - val_loss: 14939295744.0000 - val_r2: 0.4546
Epoch 3/50
337/342 [============================>.] - ETA: 0s - loss: 7962473984.0000 - r2: 0.6631
Epoch 3: val_loss improved from 14939295744.00000 to 14265612288.00000, saving model to
PartB_bestepoch\
342/342 [==============================] - 3s 7ms/step - loss: 7957949952.0000 - r2: 0.
6635 - val_loss: 14265612288.0000 - val_r2: 0.4781
Epoch 4/50
333/342 [============================>.] - ETA: 0s - loss: 7394527232.0000 - r2: 0.6863
Epoch 4: val_loss did not improve from 14265612288.00000
342/342 [==============================] - 3s 7ms/step - loss: 7381860352.0000 - r2: 0.
6869 - val_loss: 14277165056.0000 - val_r2: 0.4776
Epoch 5/50
342/342 [==============================] - ETA: 0s - loss: 6953206272.0000 - r2: 0.7049
Epoch 5: val_loss improved from 14265612288.00000 to 12391053312.00000, saving model to
PartB_bestepoch\
342/342 [==============================] - 3s 6ms/step - loss: 6953206272.0000 - r2: 0.
7049 - val_loss: 12391053312.0000 - val_r2: 0.5462
Epoch 6/50
325/342 [===========================>..] - ETA: 0s - loss: 6554208256.0000 - r2: 0.7220
Epoch 6: val_loss did not improve from 12391053312.00000
342/342 [==============================] - 2s 4ms/step - loss: 6552152576.0000 - r2: 0.
7225 - val_loss: 12759575552.0000 - val_r2: 0.5335
Epoch 7/50
340/342 [============================>.] - ETA: 0s - loss: 6162079232.0000 - r2: 0.7385
Epoch 7: val_loss improved from 12391053312.00000 to 12358235136.00000, saving model to
PartB_bestepoch\
342/342 [==============================] - 2s 5ms/step - loss: 6161995264.0000 - r2: 0.
7387 - val_loss: 12358235136.0000 - val_r2: 0.5481
Epoch 8/50
340/342 [============================>.] - ETA: 0s - loss: 5791224320.0000 - r2: 0.7545
Epoch 8: val_loss improved from 12358235136.00000 to 11545470976.00000, saving model to
PartB_bestepoch\
342/342 [==============================] - 2s 5ms/step - loss: 5791682560.0000 - r2: 0.
7543 - val_loss: 11545470976.0000 - val_r2: 0.5775
Epoch 9/50
332/342 [============================>.] - ETA: 0s - loss: 5468476928.0000 - r2: 0.7680
Epoch 9: val_loss did not improve from 11545470976.00000
342/342 [==============================] - 2s 4ms/step - loss: 5466444288.0000 - r2: 0.
7683 - val_loss: 11560046592.0000 - val_r2: 0.5764
Epoch 10/50
341/342 [============================>.] - ETA: 0s - loss: 5186738176.0000 - r2: 0.7799
Epoch 10: val_loss improved from 11545470976.00000 to 10713629696.00000, saving model t
o PartB_bestepoch\
342/342 [==============================] - 2s 5ms/step - loss: 5187721728.0000 - r2: 0.
7798 - val_loss: 10713629696.0000 - val_r2: 0.6078
Epoch 11/50
342/342 [==============================] - ETA: 0s - loss: 4964157440.0000 - r2: 0.7890
```

```
Epoch 11: val_loss did not improve from 10713629696.00000
342/342 [==============================] - 3s 7ms/step - loss: 4964157440.0000 - r2: 0.
7890 - val_loss: 11783515136.0000 - val_r2: 0.5685
Epoch 12/50
341/342 [============================>.] - ETA: 0s - loss: 4761686016.0000 - r2: 0.7979
Epoch 12: val_loss did not improve from 10713629696.00000
342/342 [==============================] - 2s 5ms/step - loss: 4760956416.0000 - r2: 0.
7980 - val_loss: 10740106240.0000 - val_r2: 0.6064
Epoch 13/50
334/342 [============================>.] - ETA: 0s - loss: 4591762944.0000 - r2: 0.8049
Epoch 13: val_loss improved from 10713629696.00000 to 10165262336.00000, saving model t
o PartB_bestepoch\
342/342 [==============================] - 2s 5ms/step - loss: 4591525888.0000 - r2: 0.
8051 - val_loss: 10165262336.0000 - val_r2: 0.6280
Epoch 14/50
339/342 [============================>.] - ETA: 0s - loss: 4438092288.0000 - r2: 0.8117
Epoch 14: val_loss did not improve from 10165262336.00000
342/342 [==============================] - 2s 4ms/step - loss: 4438109184.0000 - r2: 0.
8115 - val_loss: 11078604800.0000 - val_r2: 0.5939
Epoch 15/50
334/342 [============================>.] - ETA: 0s - loss: 4294188032.0000 - r2: 0.8174
Epoch 15: val_loss did not improve from 10165262336.00000
342/342 [==============================] - 2s 4ms/step - loss: 4298055680.0000 - r2: 0.
8175 - val_loss: 10185533440.0000 - val_r2: 0.6270
Epoch 16/50
325/342 [===========================>..] - ETA: 0s - loss: 4178732288.0000 - r2: 0.8225
Epoch 16: val_loss did not improve from 10165262336.00000
342/342 [==============================] - 2s 4ms/step - loss: 4182443776.0000 - r2: 0.
8225 - val_loss: 10620626944.0000 - val_r2: 0.6109
Epoch 17/50
340/342 [============================>.] - ETA: 0s - loss: 4078453760.0000 - r2: 0.8269
Epoch 17: val_loss improved from 10165262336.00000 to 10087306240.00000, saving model t
o PartB_bestepoch\
342/342 [==============================] - 2s 5ms/step - loss: 4080829952.0000 - r2: 0.
8267 - val_loss: 10087306240.0000 - val_r2: 0.6298
Epoch 18/50
330/342 [===========================>..] - ETA: 0s - loss: 4002146816.0000 - r2: 0.8300
Epoch 18: val_loss did not improve from 10087306240.00000
342/342 [==============================] - 2s 4ms/step - loss: 3997183488.0000 - r2: 0.
8299 - val_loss: 10753428480.0000 - val_r2: 0.6065
Epoch 19/50
340/342 [============================>.] - ETA: 0s - loss: 3938508800.0000 - r2: 0.8328
Epoch 19: val_loss did not improve from 10087306240.00000
342/342 [==============================] - 2s 4ms/step - loss: 3937834752.0000 - r2: 0.
8329 - val_loss: 10312682496.0000 - val_r2: 0.6220
Epoch 20/50
339/342 [============================>.] - ETA: 0s - loss: 3882119424.0000 - r2: 0.8351
Epoch 20: val_loss improved from 10087306240.00000 to 9415782400.00000, saving model to
PartB_bestepoch\
342/342 [==============================] - 2s 5ms/step - loss: 3881797376.0000 - r2: 0.
8352 - val_loss: 9415782400.0000 - val_r2: 0.6546
Epoch 21/50
342/342 [==============================] - ETA: 0s - loss: 3840475904.0000 - r2: 0.8366
Epoch 21: val_loss did not improve from 9415782400.00000
342/342 [==============================] - 2s 4ms/step - loss: 3840475904.0000 - r2: 0.
8366 - val_loss: 10388734976.0000 - val_r2: 0.6190
Epoch 22/50
337/342 [============================>.] - ETA: 0s - loss: 3803442176.0000 - r2: 0.8385
Epoch 22: val_loss did not improve from 9415782400.00000
342/342 [==============================] - 2s 4ms/step - loss: 3806304256.0000 - r2: 0.
```

```
8385 - val_loss: 9441271808.0000 - val_r2: 0.6543
Epoch 23/50
334/342 [============================>.] - ETA: 0s - loss: 3791863040.0000 - r2: 0.8389
Epoch 23: val_loss did not improve from 9415782400.00000
342/342 [==============================] - 2s 4ms/step - loss: 3792311808.0000 - r2: 0.
8390 - val_loss: 9438312448.0000 - val_r2: 0.6544
Epoch 24/50
333/342 [============================>.] - ETA: 0s - loss: 3774094592.0000 - r2: 0.8398
Epoch 24: val_loss did not improve from 9415782400.00000
342/342 [==============================] - 2s 4ms/step - loss: 3770861312.0000 - r2: 0.
8396 - val_loss: 10380033024.0000 - val_r2: 0.6199
Epoch 25/50
340/342 [============================>.] - ETA: 0s - loss: 3758390016.0000 - r2: 0.8406
Epoch 25: val_loss did not improve from 9415782400.00000
342/342 [==============================] - 2s 4ms/step - loss: 3759707392.0000 - r2: 0.
8404 - val_loss: 10070590464.0000 - val_r2: 0.6311
Epoch 26/50
341/342 [============================>.] - ETA: 0s - loss: 3738425088.0000 - r2: 0.8412
Epoch 26: val_loss did not improve from 9415782400.00000
342/342 [==============================] - 2s 5ms/step - loss: 3738257152.0000 - r2: 0.
8410 - val_loss: 10189365248.0000 - val_r2: 0.6271
Epoch 27/50
337/342 [============================>.] - ETA: 0s - loss: 3724519680.0000 - r2: 0.8417
Epoch 27: val_loss did not improve from 9415782400.00000
342/342 [==============================] - 2s 4ms/step - loss: 3730751744.0000 - r2: 0.
8416 - val_loss: 9697868800.0000 - val_r2: 0.6442
Epoch 28/50
340/342 [============================>.] - ETA: 0s - loss: 3726867200.0000 - r2: 0.8416
Epoch 28: val_loss did not improve from 9415782400.00000
342/342 [==============================] - 2s 4ms/step - loss: 3726139904.0000 - r2: 0.
8414 - val_loss: 10179208192.0000 - val_r2: 0.6268
Epoch 29/50
341/342 [============================>.] - ETA: 0s - loss: 3716480000.0000 - r2: 0.8423
Epoch 29: val_loss improved from 9415782400.00000 to 9402954752.00000, saving model to
PartB_bestepoch\
342/342 [==============================] - 2s 5ms/step - loss: 3716092416.0000 - r2: 0.
8424 - val_loss: 9402954752.0000 - val_r2: 0.6555
Epoch 30/50
336/342 [============================>.] - ETA: 0s - loss: 3708004608.0000 - r2: 0.8426
Epoch 30: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 5ms/step - loss: 3711296512.0000 - r2: 0.
8421 - val_loss: 10483511296.0000 - val_r2: 0.6156
Epoch 31/50
332/342 [============================>.] - ETA: 0s - loss: 3703522560.0000 - r2: 0.8423
Epoch 31: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3702434048.0000 - r2: 0.
8425 - val_loss: 9969513472.0000 - val_r2: 0.6349
Epoch 32/50
333/342 [============================>.] - ETA: 0s - loss: 3706041856.0000 - r2: 0.8429
Epoch 32: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3697982208.0000 - r2: 0.
8430 - val_loss: 10640462848.0000 - val_r2: 0.6102
Epoch 33/50
339/342 [============================>.] - ETA: 0s - loss: 3689257984.0000 - r2: 0.8433
Epoch 33: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3690035456.0000 - r2: 0.
8432 - val_loss: 9882433536.0000 - val_r2: 0.6381
Epoch 34/50
341/342 [============================>.] - ETA: 0s - loss: 3685798912.0000 - r2: 0.8434
Epoch 34: val_loss did not improve from 9402954752.00000
```

```
342/342 [==============================] - 2s 4ms/step - loss: 3686131968.0000 - r2: 0.
8434 - val_loss: 10364608512.0000 - val_r2: 0.6199
Epoch 35/50
335/342 [============================>.] - ETA: 0s - loss: 3688043776.0000 - r2: 0.8432
Epoch 35: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3683887616.0000 - r2: 0.
8433 - val_loss: 9852348416.0000 - val_r2: 0.6392
Epoch 36/50
342/342 [==============================] - ETA: 0s - loss: 3682795008.0000 - r2: 0.8435
Epoch 36: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3682795008.0000 - r2: 0.
8435 - val_loss: 9692525568.0000 - val_r2: 0.6449
Epoch 37/50
338/342 [============================>.] - ETA: 0s - loss: 3676841472.0000 - r2: 0.8437
Epoch 37: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3678804224.0000 - r2: 0.
8434 - val_loss: 10471049216.0000 - val_r2: 0.6154
Epoch 38/50
325/342 [===========================>..] - ETA: 0s - loss: 3689175808.0000 - r2: 0.8432
Epoch 38: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3679342336.0000 - r2: 0.
8436 - val_loss: 10943375360.0000 - val_r2: 0.5992
Epoch 39/50
342/342 [==============================] - ETA: 0s - loss: 3670298368.0000 - r2: 0.8444
Epoch 39: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3670298368.0000 - r2: 0.
8444 - val_loss: 10448453632.0000 - val_r2: 0.6163
Epoch 40/50
336/342 [============================>.] - ETA: 0s - loss: 3674622208.0000 - r2: 0.8440
Epoch 40: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3671521536.0000 - r2: 0.
8442 - val_loss: 9807437824.0000 - val_r2: 0.6404
Epoch 41/50
335/342 [============================>.] - ETA: 0s - loss: 3662637568.0000 - r2: 0.8445
Epoch 41: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3663005184.0000 - r2: 0.
8444 - val_loss: 10278647808.0000 - val_r2: 0.6227
Epoch 42/50
340/342 [============================>.] - ETA: 0s - loss: 3670224640.0000 - r2: 0.8440
Epoch 42: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3669022208.0000 - r2: 0.
8439 - val_loss: 10889697280.0000 - val_r2: 0.6006
Epoch 43/50
336/342 [============================>.] - ETA: 0s - loss: 3664304896.0000 - r2: 0.8442
Epoch 43: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3661965312.0000 - r2: 0.
8442 - val_loss: 10192940032.0000 - val_r2: 0.6266
Epoch 44/50
331/342 [============================>.] - ETA: 0s - loss: 3662964224.0000 - r2: 0.8441
Epoch 44: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3663009024.0000 - r2: 0.
8444 - val_loss: 10419182592.0000 - val_r2: 0.6186
Epoch 45/50
342/342 [==============================] - ETA: 0s - loss: 3655868160.0000 - r2: 0.8445
Epoch 45: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3655868160.0000 - r2: 0.
8445 - val_loss: 10176596992.0000 - val_r2: 0.6265
Epoch 46/50
335/342 [============================>.] - ETA: 0s - loss: 3662997248.0000 - r2: 0.8446
Epoch 46: val_loss did not improve from 9402954752.00000
```

```
342/342 [==============================] - 2s 4ms/step - loss: 3657720064.0000 - r2: 0.
8446 - val_loss: 10224751616.0000 - val_r2: 0.6250
Epoch 47/50
338/342 [============================>.] - ETA: 0s - loss: 3657480448.0000 - r2: 0.8445
Epoch 47: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3657137408.0000 - r2: 0.
8446 - val_loss: 10177386496.0000 - val_r2: 0.6276
Epoch 48/50
333/342 [============================>.] - ETA: 0s - loss: 3652964352.0000 - r2: 0.8447
Epoch 48: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3653188608.0000 - r2: 0.
8448 - val_loss: 10679427072.0000 - val_r2: 0.6079
Epoch 49/50
335/342 [============================>.] - ETA: 0s - loss: 3656343552.0000 - r2: 0.8446
Epoch 49: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3656990720.0000 - r2: 0.
8446 - val_loss: 10527970304.0000 - val_r2: 0.6138
Epoch 50/50
336/342 [============================>.] - ETA: 0s - loss: 3646724608.0000 - r2: 0.8450
Epoch 50: val_loss did not improve from 9402954752.00000
342/342 [==============================] - 2s 4ms/step - loss: 3652719616.0000 - r2: 0.
8447 - val_loss: 10181057536.0000 - val_r2: 0.6268
INFO:tensorflow:Assets written to: PartB_best_model/assets
```

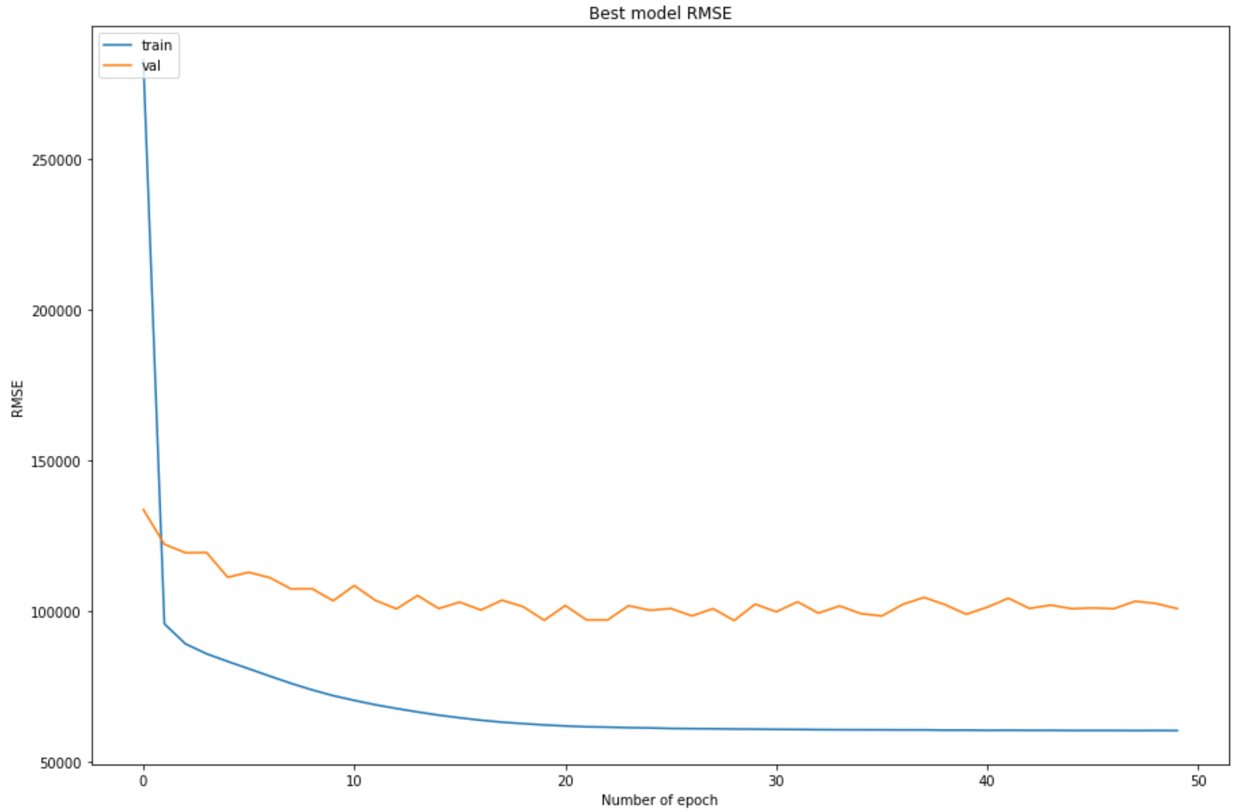
In [ ]:

```python
def square_roots(l):
    result = [sqrt(i) for i in l]
    return result
```

Plot of Train & Test RMSE for 50 epochs

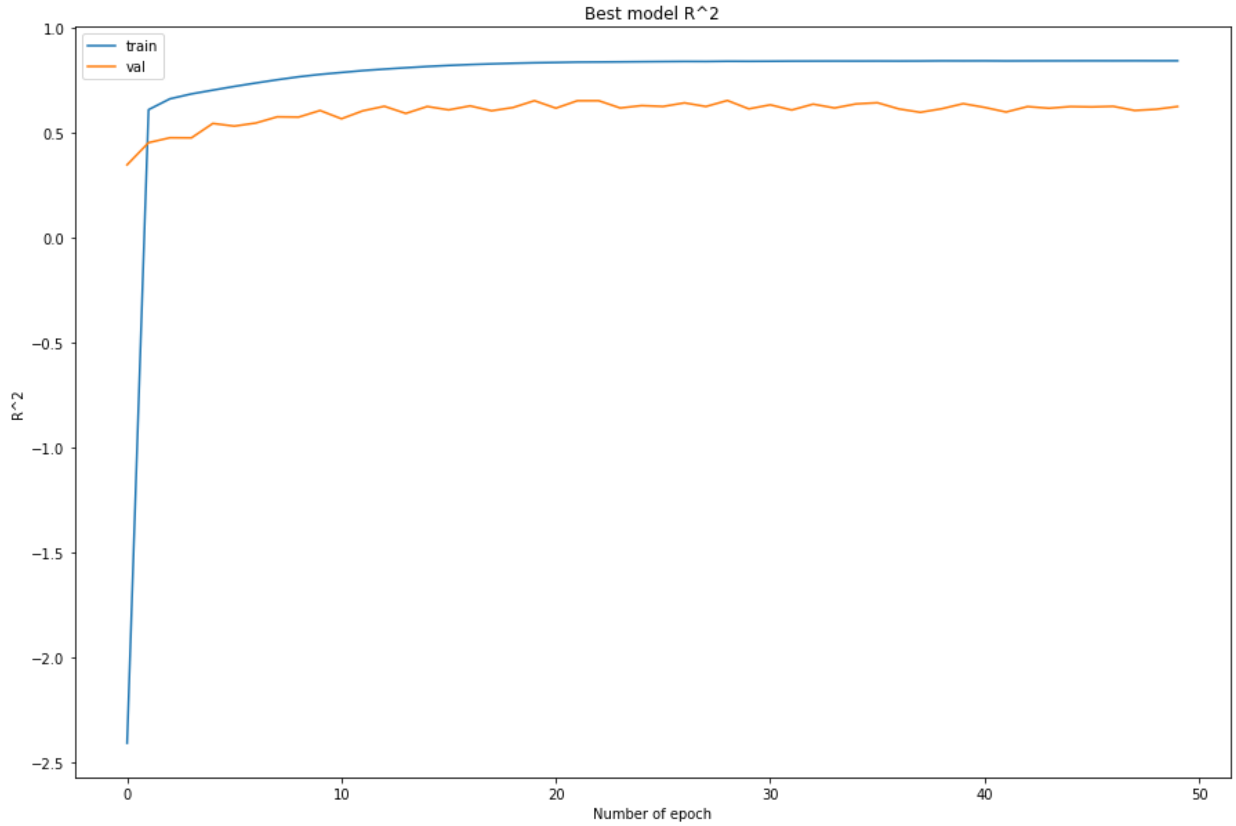
In [ ]:

```python
import matplotlib.pyplot as plt
plt_1 = plt.figure(figsize=(15, 10))
# Plot
plt.plot(square_roots(Q2_best_model_history['best_model'].history['loss']))
plt.plot(square_roots(Q2_best_model_history['best_model'].history['val_loss']))
plt.title('Best model RMSE')
plt.ylabel('RMSE')
plt.xlabel('Number of epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



Plot of Train & Test R^2 for 50 epochs

In [ ]:
```
plt_1 = plt.figure(figsize=(15, 10))
plt.plot(Q2_best_model_history['best_model'].history['r2'])
plt.plot(Q2_best_model_history['best_model'].history['val_r2'])
plt.title('Best model R^2')
plt.ylabel('R^2')
plt.xlabel('Number of epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



Question 2E

In [ ]:

```
# Do a prediction on test set and look for error against predicted - actual


def flatten(l):
    return [item for sublist in l for item in sublist]

def df_to_dataset(dataframe):
    dataframe = dataframe.copy()

    labels = dataframe.pop("resale_price")
    ds = tf.data.Dataset.from_tensor_slices((dict(dataframe), labels))
    return ds

## We use the test dataframe defined in the first question (Test > 2020)
Q2E_test_df = test_dataframe.copy()
Q2E_test_ds = df_to_dataset(Q2E_test_df)
Q2E_test_ds = Q2E_test_ds.batch(256)

# Load the best weights
Q2_best_model.load_weights('PartB_bestepoch/')

prediction =  Q2_best_model.predict(Q2E_test_ds)
prediction = flatten(prediction)
data_df = test_dataframe.copy()
data_df["Predicted Resale Value"] = prediction

data_df["Error"] = abs(data_df["resale_price"] - data_df["Predicted Resale Value"])
data_df = data_df.sort_values(by="Error", ascending=False)
data_df = data_df.head(30)
```

```
data_df
```

```
 15/180 [=>............................] - ETA: 0s
```

```
c:\Users\JoeTe\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\engine\functional.py:566: UserWarning: Input dict contained keys ['year'] which did not match any model input. They will be ignored by the model.
  inputs = self._flatten_to_reference_inputs(inputs)
```

```
180/180 [==============================] - 1s 4ms/step
```

Out[ ]:

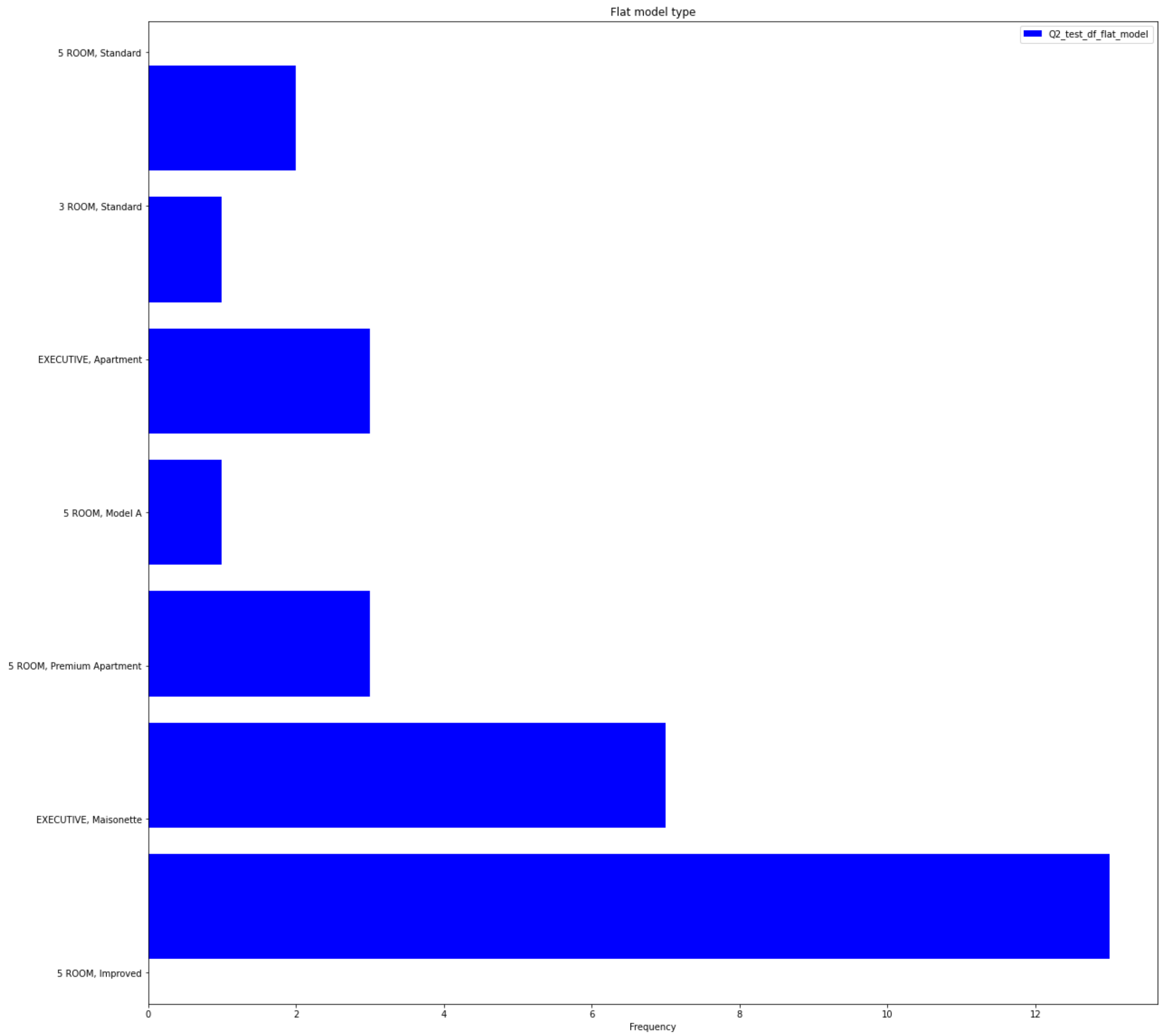
	month	year	dist_to_nearest_stn	dist_to_dhoby	degree_centrality	eigenvector_centrality	flat
119399	5	2022	0.586629	2.932814	0.016807	0.047782	
114504	12	2021	0.428356	8.948410	0.016807	0.001358	
127227	8	2022	0.489478	3.977493	0.016807	0.008342	
119400	5	2022	0.586629	2.932814	0.016807	0.047782	
120164	1	2022	0.271583	9.003026	0.016807	0.001358	
121586	4	2022	0.245502	9.313260	0.016807	0.001179	
127207	8	2022	0.504800	5.727076	0.016807	0.010276	5 R
120166	3	2022	0.271583	9.003026	0.016807	0.001358	
117107	7	2022	1.216557	8.071776	0.016807	0.006243	
127251	8	2022	0.334556	5.561626	0.016807	0.010276	
114505	12	2021	0.473544	8.936025	0.016807	0.001358	
117058	6	2022	1.722450	7.861222	0.016807	0.006243	
90353	2	2021	0.271583	9.003026	0.016807	0.001358	
111790	11	2021	0.581977	2.309477	0.016807	0.047782	
125467	5	2022	1.192284	14.877669	0.016807	0.000127	
118295	7	2022	0.786740	6.514619	0.033613	0.015854	
117054	4	2022	1.722450	7.861222	0.016807	0.006243	
121584	4	2022	0.245502	9.313260	0.016807	0.001179	
117057	5	2022	1.722450	7.861222	0.016807	0.006243	
117052	2	2022	1.722450	7.861222	0.016807	0.006243	
100263	6	2021	0.245207	4.709043	0.016807	0.008342	

	month	year	dist_to_nearest_stn	dist_to_dhoby	degree_centrality	eigenvector_centrality	flat
117060	7	2022	1.745057	7.818413	0.016807	0.006243	
127289	7	2022	0.245207	4.709043	0.016807	0.008342	
125026	8	2022	1.949971	7.761923	0.016807	0.002799	
117061	7	2022	1.722450	7.861222	0.016807	0.006243	
127226	7	2022	0.489478	3.977493	0.016807	0.008342	
124901	6	2022	0.187007	3.052121	0.016807	0.047857	
118234	7	2022	0.947205	6.663943	0.033613	0.015854	
111865	11	2021	0.686789	2.664024	0.016807	0.047782	
125006	1	2022	1.821677	6.985963	0.016807	0.007049	

Identifying the patterns and trends with a histogram plots of certain features

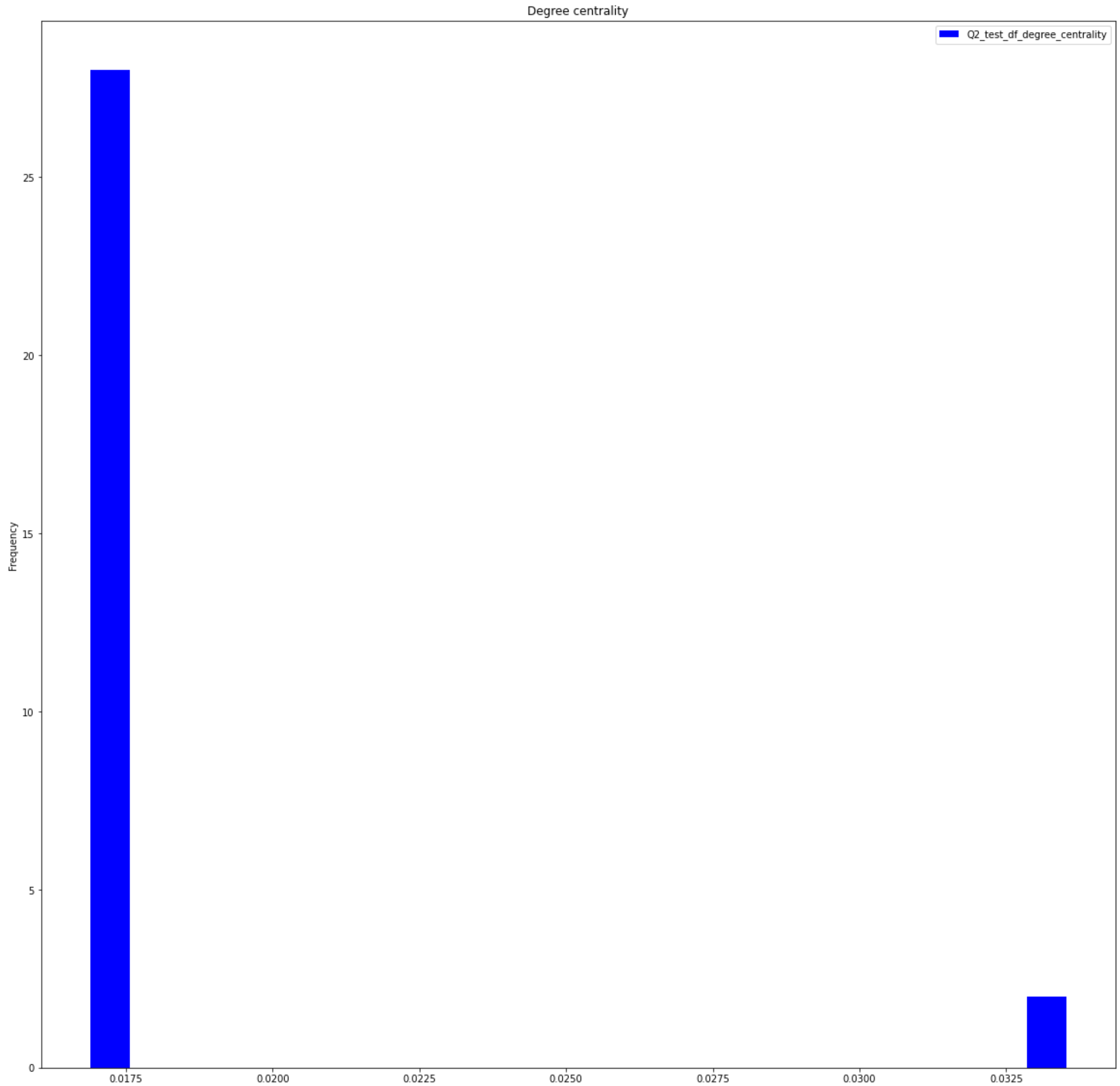
In [ ]:

```
fig, axes =plt.subplots(1, figsize=(20,20))
axes.hist(data_df['flat_model_type'],bins = data_df["flat_model_type"].nunique(), histty
axes.legend(loc='upper right')
axes.set_title('Flat model type')
axes.set_xlabel('Frequency')
plt.show()
```



In [ ]:

```
fig, axes =plt.subplots(1, figsize=(20,20))
axes.hist(data_df['degree_centrality'],bins = 20, histtype='bar', color =['blue'], label
axes.legend(loc='upper right')
axes.set_title('Degree centrality')
axes.set_ylabel('Frequency')
plt.show()
```



List down the trends and suggest a way to reduce the errors

Since the best model was train using train_ds, I will be using test_ds for the model's prediction. From the model's prediction, we can see that the largest error mainly comes from high value HDB apartment (5 room or better). This could be due to sharp rise in prices of 5 room or better HDB apartments over the past few years as compared to the other flat types.

In addition, the degree centrality mainly holds at the value = 0.016807.

One solution will be to introduce more data into the training dataset which are from the past year(2021) and current year(2022) such that the model have the information to learn about current HDB pricing trends.