pca_enclosed_office

January 18, 2022

1 Principal Component Analysis

Below are the Principal Component Analysis on the first 9 features:

1. index: Index for each data point
2. sim: Index for each simulation
3. mass: Total mass flow rate of air supply to the zone (kg/s)
4. tem_supply: Temperature of air supplied to the zone (°C)
5. tem_surf: Temperature of walls, ceiling, and floor (°C)
6. tem_win: Temperature of windows (°C)
7. n_occ: Number of occupants in zone
8. x: location of occupant of interest on x-axis (m)
9. y: location of occupant of interest on y-axis (m)

The last 3 features are treated as target predictions for evaluation purpose.

1. MRT: Mean radiant temperature of the occupant of interest's head and chest (°C)
2. T: Average temperature of air surrounding the occupant of interest (°C)
3. V: Average speed of air surrounding the occupant of interest occupant (°C)

[1]:
```
data_loc = '../data/preprocessed/'
```

1.1 Enclosed Office

1.1.1 1. Standardize the data

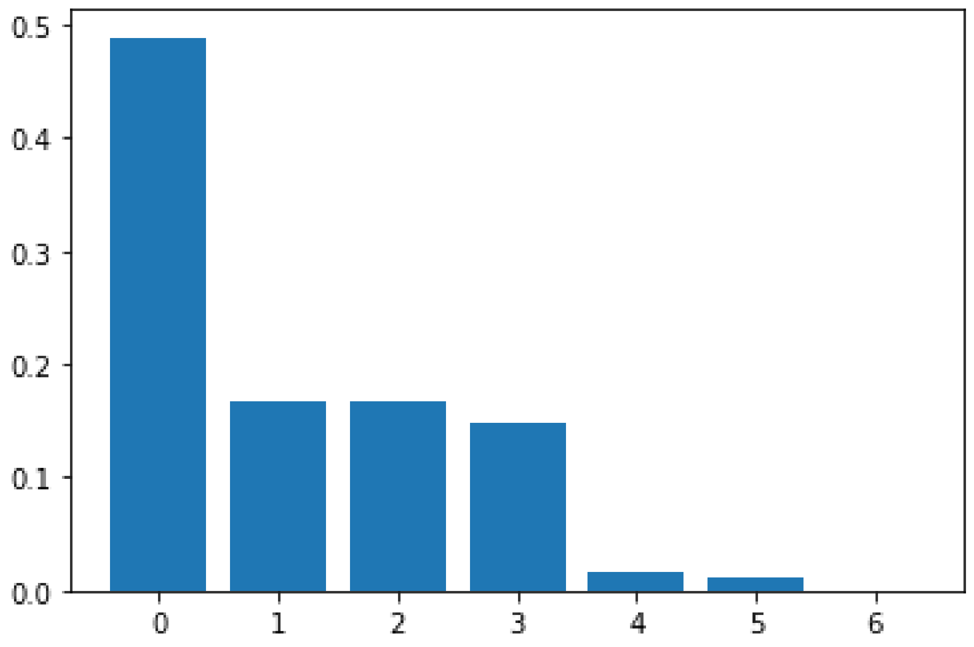
[2]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler

enclosed_office_data = pd.read_csv(data_loc+"enclosed_office.csv")

# Remove the index column
enclosed_office_data.drop(columns=['index', 'sim'], inplace=True)

# Seperate the data into features and labels
X = enclosed_office_data.drop(columns=['MRT', 'T', 'V'])
y_MRT = enclosed_office_data['MRT']
y_T = enclosed_office_data['T']
y_V = enclosed_office_data['V']
```

```
# Standardize the data
scaler = StandardScaler()
scaler.fit(X)

X_std = pd.DataFrame(scaler.transform(X), columns=X.columns)
X_std.describe()
```

[2]:
```
               mass    tem_supply      tem_surf       tem_win       n_occ  \
count  7.200000e+02  7.200000e+02  7.200000e+02  7.200000e+02  720.000000
mean  -1.102205e-15 -8.172475e-16  1.174986e-16  1.264421e-16    0.000000
std    1.000695e+00  1.000695e+00  1.000695e+00  1.000695e+00    1.000695
min   -9.597991e-01 -1.248663e+00 -1.448365e+00 -1.303552e+00   -1.000000
25%   -9.595023e-01 -9.487674e-01 -8.687326e-01 -9.304797e-01   -1.000000
50%   -4.197757e-01 -4.979886e-02  1.375536e-04  9.659024e-02    0.000000
75%    1.379265e+00  8.484505e-01  8.691235e-01  1.029944e+00    1.000000
max    1.379898e+00  1.448242e+00  1.447946e+00  1.216779e+00    1.000000

                  x           y
count  7.200000e+02  720.000000
mean  -4.440892e-16    0.000000
std    0.000000e+00    1.000695
min   -4.440892e-16   -1.000000
25%   -4.440892e-16   -1.000000
50%   -4.440892e-16    0.000000
75%   -4.440892e-16    1.000000
max   -4.440892e-16    1.000000
```

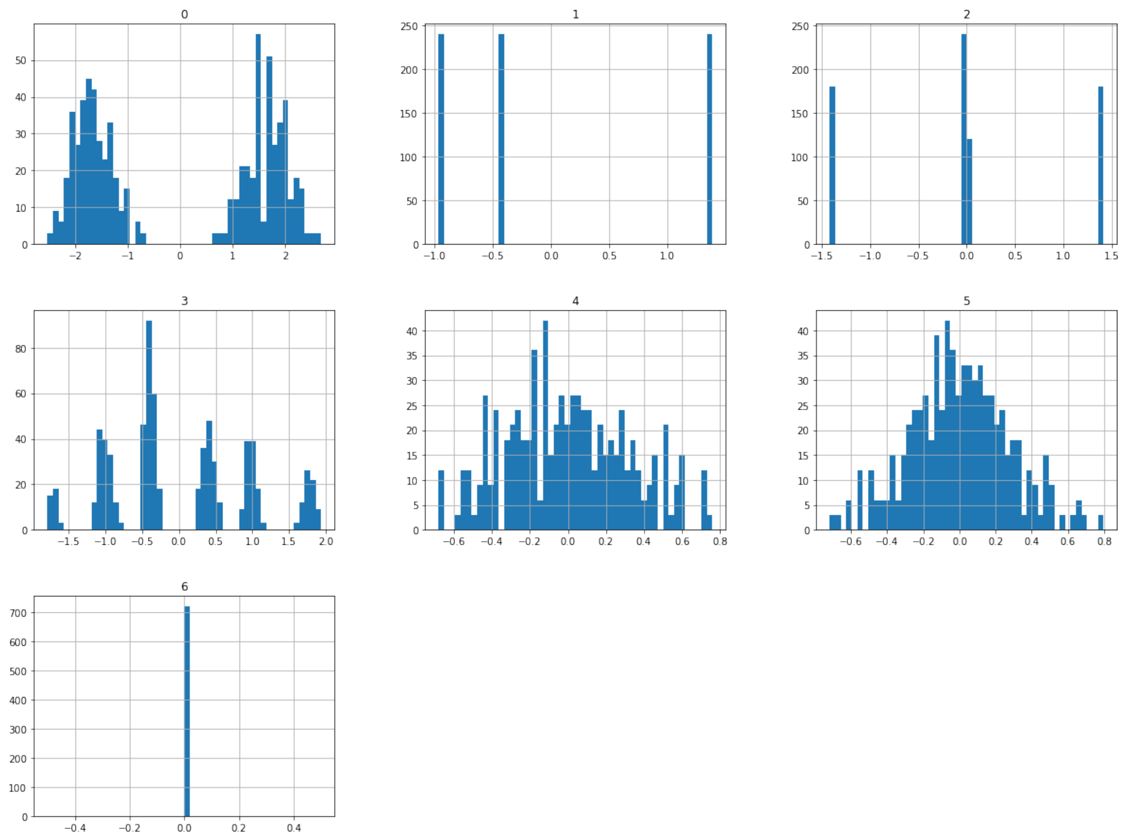
1.1.2 2. Perform PCA

[3]:
```
# Perform PCA
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np
from numpy import savetxt

pca = PCA()
pca.fit(X_std)

e_vectors = pca.components_ # The eigenvectors
evr = pca.explained_variance_ratio_ # The variance explained by each eigenvector

# Save the eigenvectors to a text file
savetxt("../reports/data/pca_enclosed_office_eigenvectors.csv", e_vectors,␣
 ↪delimiter=',')

plt.bar(range(len(evr)), evr)
plt.savefig('../figures/evr_pca_enclosed_office.png')
```



[4]:
```
# Tramsform the data
X_pca = pd.DataFrame(pca.transform(X_std))
fig, ax = plt.subplots(figsize=(20,15))
X_pca.hist(bins=50, ax=ax)
fig.savefig('../figures/hist_pca_enclosed_office.png')
```

```
/tmp/ipykernel_9808/3863644822.py:4: UserWarning: To output multiple subplots,
the figure containing the passed axes is being cleared
  X_pca.hist(bins=50, ax=ax)
```



1.1.3 3. Compare models' results with and without PCA on predicting MRT

3.1 Split the data into training and testing

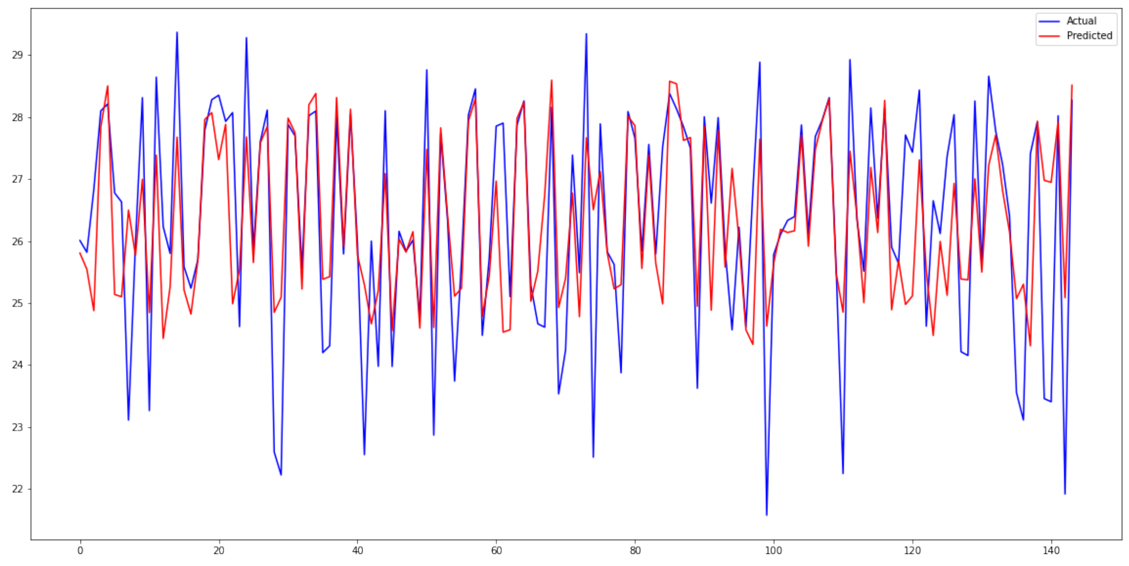
[5]:
```python
# Split the data into training and testing
from sklearn.model_selection import train_test_split

# Without PCA
X_train, X_test, y_train, y_test = train_test_split(X_std, y_MRT, test_size=0.
↪2, random_state=42)

# With PCA
X_train_pca, X_test_pca, y_train_pca, y_test_pca = train_test_split(X_pca,␣
↪y_MRT, test_size=0.2, random_state=42)
```

3.2 Train and evaluate the Linear Regression model

3.2.1 Without PCA

[6]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pickle
import time

# Train linear regression model on training data
lr_model = LinearRegression()
start = time.time()
lr_model.fit(X_train, y_train)
stop = time.time()
lr_train_time = stop - start
print(f"Training time: {lr_train_time}s")

# Save the model to a pickle file
filename = '../reports/models/lr_MRT_enclosed_office_model.pkl'
pickle.dump(lr_model, open(filename, 'wb'))

# Predict on test data
y_pred = lr_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

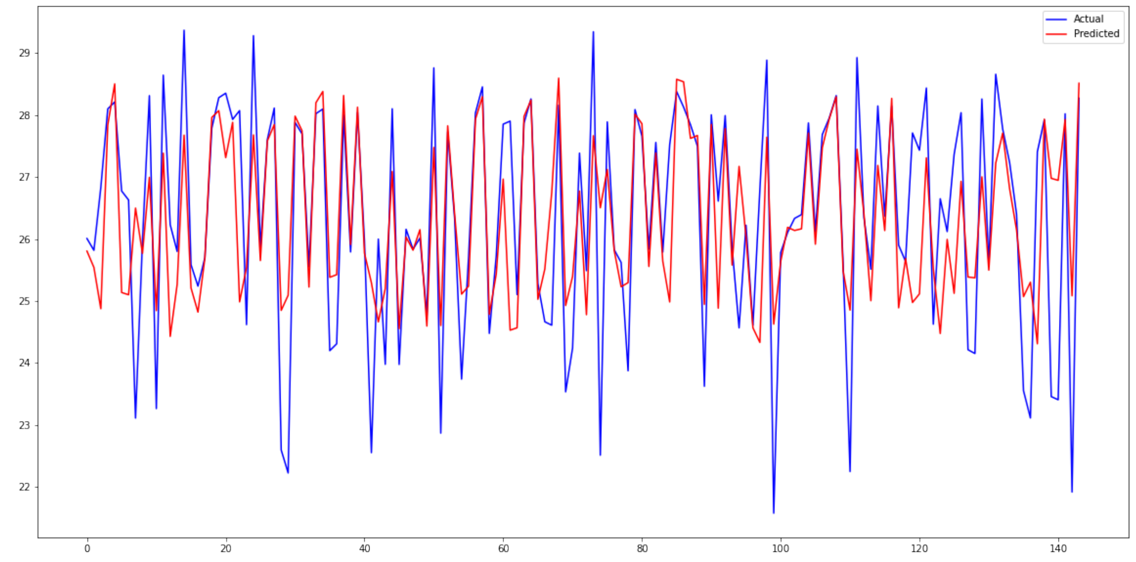
```
Training time: 0.0037326812744140625s
Mean squared error: 1.7880454889062574
```



3.2.2 With PCA

[7]:
```python
# Train linear regression model on training data
start = time.time()
lr_model.fit(X_train_pca, y_train_pca)
stop = time.time()
lr_pca_train_time = stop - start
print(f"Training time: {lr_pca_train_time}s")

# Predict on test data
y_pred_pca = lr_model.predict(X_test_pca)

# Save the model to a pickle file
filename = '../reports/models/lr_MRT_pca_enclosed_office_model.pkl'
pickle.dump(lr_model, open(filename, 'wb'))

# Evaluate the model
mse = mean_squared_error(y_test_pca, y_pred_pca)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test_pca[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred_pca[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

```
Training time: 0.00335693359375s
Mean squared error: 1.7880454889062571
```



3.3 Train and evaluate the Feedforward Neural Network model

3.2.1 Without PCA

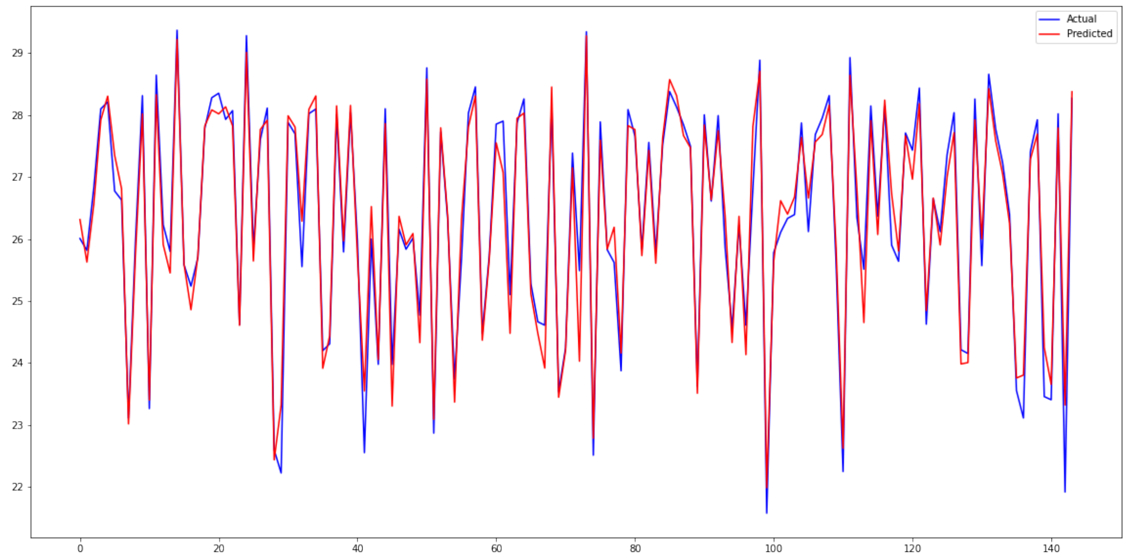
[8]:
```
# Create neural network model
from keras.models import Sequential
from keras.layers import Dense

N_NEURONS = 1024
N_LAYERS = 4

nn_model = Sequential()
nn_model.add(Dense(units=N_NEURONS, input_dim=X.shape[1], activation='relu'))
for i in range(N_LAYERS-1):
    nn_model.add(Dense(units=N_NEURONS, activation='relu'))
nn_model.add(Dense(units=1, activation='linear')) # Output layer
nn_model.compile(loss='mean_squared_error', optimizer='adam')

# Train the model
start = time.time()
nn_model.fit(X_train, y_train, epochs=100, verbose=0)
stop = time.time()
nn_train_time = stop - start
print(f"Training time: {nn_train_time}s")

# Predict on test data
y_pred_nn = nn_model.predict(X_test)
nn_model.save('../reports/models/nn_MRT_enclosed_office_model.pkl')

# Evaluate the model
mse = mean_squared_error(y_test, y_pred_nn)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred_nn[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

```
2022-01-18 15:42:08.606725: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcudart.so.10.1
2022-01-18 15:42:12.222488: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
```

```
opened dynamic library libcuda.so.1
2022-01-18 15:42:12.313180: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-01-18 15:42:12.313795: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1716] Found device 0 with
properties:
pciBusID: 0000:04:00.0 name: GeForce GTX 1060 6GB computeCapability: 6.1
coreClock: 1.7845GHz coreCount: 10 deviceMemorySize: 5.93GiB
deviceMemoryBandwidth: 178.99GiB/s
2022-01-18 15:42:12.313845: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcudart.so.10.1
2022-01-18 15:42:12.450401: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcublas.so.10
2022-01-18 15:42:12.472239: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcufft.so.10
2022-01-18 15:42:12.480658: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcurand.so.10
2022-01-18 15:42:12.514768: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcusolver.so.10
2022-01-18 15:42:12.525988: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcusparse.so.10
2022-01-18 15:42:12.682947: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcudnn.so.7
2022-01-18 15:42:12.683269: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-01-18 15:42:12.684430: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-01-18 15:42:12.685335: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1858] Adding visible gpu
devices: 0
2022-01-18 15:42:12.726762: I
tensorflow/core/platform/profile_utils/cpu_utils.cc:104] CPU Frequency:
2194700000 Hz
2022-01-18 15:42:12.730188: I tensorflow/compiler/xla/service/service.cc:168]
XLA service 0x5637f4fa2e10 initialized for platform Host (this does not
guarantee that XLA will be used). Devices:
2022-01-18 15:42:12.730244: I tensorflow/compiler/xla/service/service.cc:176]
StreamExecutor device (0): Host, Default Version
2022-01-18 15:42:13.018653: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-01-18 15:42:13.019407: I tensorflow/compiler/xla/service/service.cc:168]
XLA service 0x5637f5013010 initialized for platform CUDA (this does not
guarantee that XLA will be used). Devices:
2022-01-18 15:42:13.019454: I tensorflow/compiler/xla/service/service.cc:176]
StreamExecutor device (0): GeForce GTX 1060 6GB, Compute Capability 6.1
2022-01-18 15:42:13.021743: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-01-18 15:42:13.022748: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1716] Found device 0 with
properties:
pciBusID: 0000:04:00.0 name: GeForce GTX 1060 6GB computeCapability: 6.1
coreClock: 1.7845GHz coreCount: 10 deviceMemorySize: 5.93GiB
deviceMemoryBandwidth: 178.99GiB/s
2022-01-18 15:42:13.022861: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcudart.so.10.1
2022-01-18 15:42:13.022950: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcublas.so.10
2022-01-18 15:42:13.023027: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcufft.so.10
2022-01-18 15:42:13.023103: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcurand.so.10
2022-01-18 15:42:13.023185: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcusolver.so.10
2022-01-18 15:42:13.023263: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcusparse.so.10
2022-01-18 15:42:13.023340: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcudnn.so.7
2022-01-18 15:42:13.023538: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-01-18 15:42:13.024632: I
```

```
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-01-18 15:42:13.025524: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1858] Adding visible gpu
devices: 0
2022-01-18 15:42:13.026656: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcudart.so.10.1
2022-01-18 15:42:16.420394: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1257] Device interconnect
StreamExecutor with strength 1 edge matrix:
2022-01-18 15:42:16.420452: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1263]      0
2022-01-18 15:42:16.420472: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1276] 0:   N
2022-01-18 15:42:16.422804: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-01-18 15:42:16.423457: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-01-18 15:42:16.424009: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1402] Created TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:0 with 4708 MB memory) -> physical
GPU (device: 0, name: GeForce GTX 1060 6GB, pci bus id: 0000:04:00.0, compute
capability: 6.1)
2022-01-18 15:42:17.705791: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcublas.so.10

Training time: 8.631425857543945s
WARNING:tensorflow:From /home/khiem/anaconda3/lib/python3.8/site-
packages/tensorflow/python/training/tracking/tracking.py:111:
Model.state_updates (from tensorflow.python.keras.engine.training) is deprecated
and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied
automatically.
WARNING:tensorflow:From /home/khiem/anaconda3/lib/python3.8/site-
packages/tensorflow/python/training/tracking/tracking.py:111: Layer.updates
(from tensorflow.python.keras.engine.base_layer) is deprecated and will be
removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied
automatically.
```

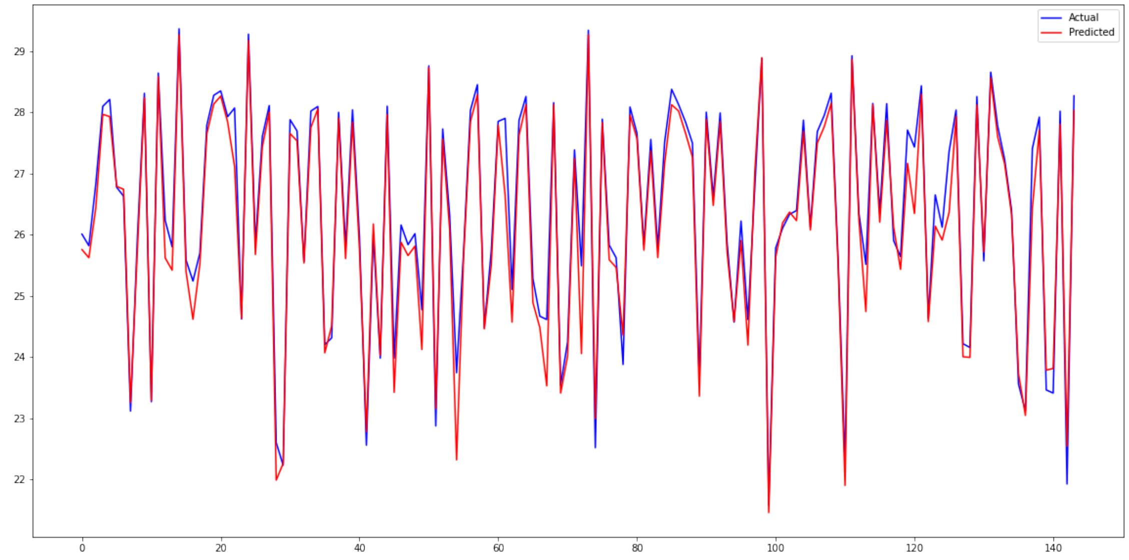
```
2022-01-18 15:42:25.887882: W tensorflow/python/util/util.cc:348] Sets are not
currently considered sequences, but this may change in the future, so consider
avoiding using them.

INFO:tensorflow:Assets written to:
../reports/models/nn_MRT_enclosed_office_model.pkl/assets
Mean squared error: 0.14294993546937929
```



3.3.2 With PCA

[9]:
```python
# Train the model
start = time.time()
nn_model.fit(X_train_pca, y_train_pca, epochs=100, verbose=0)
stop = time.time()
nn_pca_train_time = stop - start
print(f"Training time: {nn_pca_train_time}s")

# Predict on test data
y_pred_nn_pca = nn_model.predict(X_test_pca)
nn_model.save('../reports/models/nn_MRT_pca_enclosed_office_model.pkl')

# Evaluate the model
mse = mean_squared_error(y_test_pca, y_pred_nn_pca)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test_pca[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred_nn_pca[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

```
Training time: 6.020442724227905s
INFO:tensorflow:Assets written to:
../reports/models/nn_MRT_pca_enclosed_office_model.pkl/assets
Mean squared error: 0.13365892240167718
```



[10]:
```
# Save training times to a csv file
import csv

with open('../reports/data/training_times_enclosed_office_MRT.csv', 'w') as␣
↪csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Model', 'Training time (s)'])
    writer.writerow(['Linear regression', lr_train_time])
    writer.writerow(['Linear regression with PCA', lr_pca_train_time])
    writer.writerow(['Neural network', nn_train_time])
    writer.writerow(['Neural network with PCA', nn_pca_train_time])
```

1.1.4 4. Compare models' results with and without PCA on predicting T

4.1 Split the data into training and testing

[11]:
```
# Split the data into training and testing
from sklearn.model_selection import train_test_split

# Without PCA
X_train, X_test, y_train, y_test = train_test_split(X_std, y_T, test_size=0.2,␣
↪random_state=42)
```

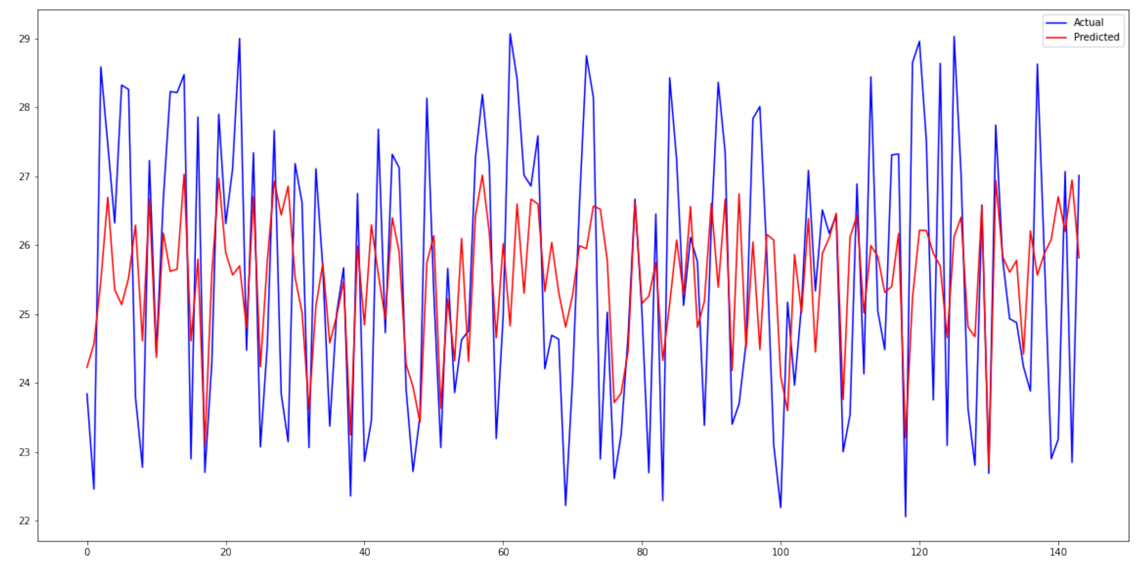
```
# With PCA
X_train_pca, X_test_pca, y_train_pca, y_test_pca = train_test_split(X_pca, y_T,␣
↪test_size=0.2, random_state=42)
```

4.2 Train and evaluate the Linear Regression model

4.2.1 Without PCA

[12]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pickle
import time

# Train linear regression model on training data
start = time.time()
lr_model.fit(X_train, y_train)
stop = time.time()
lr_train_time = stop - start
print(f"Training time: {lr_train_time}s")

# Save the model to a pickle file
filename = '../reports/models/lr_T_enclosed_office_model.pkl'
pickle.dump(lr_model, open(filename, 'wb'))

# Predict on test data
y_pred = lr_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

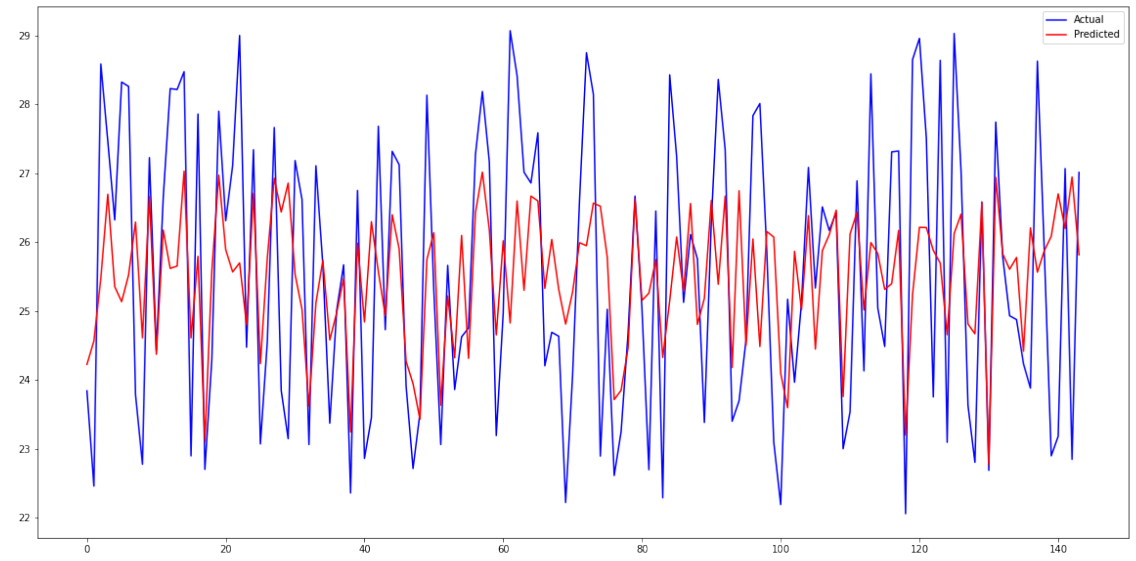
```
Training time: 0.005155633544921875s
Mean squared error: 3.0035470408263496
```



4.2.2 With PCA

[13]:
```python
# Train linear regression model on training data
start = time.time()
lr_model.fit(X_train_pca, y_train_pca)
stop = time.time()
lr_pca_train_time = stop - start
print(f"Training time: {lr_pca_train_time}s")

# Predict on test data
y_pred_pca = lr_model.predict(X_test_pca)

# Save the model to a pickle file
filename = '../reports/models/lr_T_pca_enclosed_office_model.pkl'
pickle.dump(lr_model, open(filename, 'wb'))

# Evaluate the model
mse = mean_squared_error(y_test_pca, y_pred_pca)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test_pca[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred_pca[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

```
Training time: 0.0040874481201171875s
Mean squared error: 3.00354704082635
```

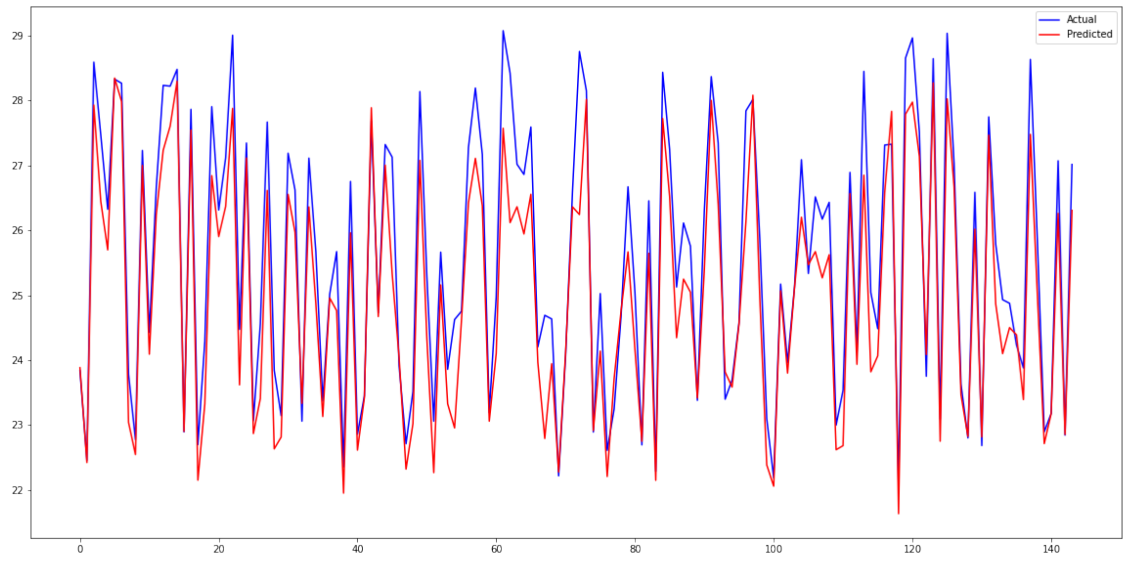


4.3 Train and evaluate the Feedforward Neural Network model

4.2.1 Without PCA

[14]:
```
# Create neural network model
from keras.models import Sequential
from keras.layers import Dense

# Train the model
start = time.time()
nn_model.fit(X_train, y_train, epochs=100, verbose=0)
stop = time.time()
nn_train_time = stop - start
print(f"Training time: {nn_train_time}s")

# Predict on test data
y_pred_nn = nn_model.predict(X_test)
nn_model.save('../reports/models/nn_T_enclosed_office_model.pkl')

# Evaluate the model
mse = mean_squared_error(y_test, y_pred_nn)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred_nn[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

```
Training time: 6.271856784820557s
INFO:tensorflow:Assets written to:
../reports/models/nn_T_enclosed_office_model.pkl/assets
Mean squared error: 0.5687313043223633
```

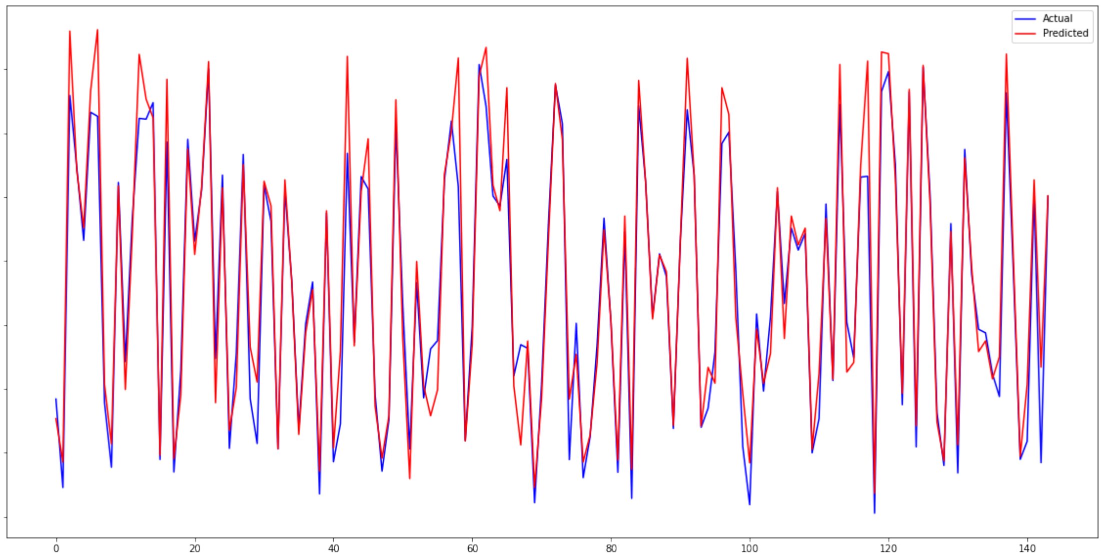


4.3.2 With PCA

[15]:
```python
# Train the model
start = time.time()
nn_model.fit(X_train_pca, y_train_pca, epochs=100, verbose=0)
stop = time.time()
nn_pca_train_time = stop - start
print(f"Training time: {nn_pca_train_time}s")

# Predict on test data
y_pred_nn_pca = nn_model.predict(X_test_pca)
nn_model.save('../reports/models/nn_T_pca_enclosed_office_model.pkl')

# Evaluate the model
mse = mean_squared_error(y_test_pca, y_pred_nn_pca)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test_pca[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred_nn_pca[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

```
Training time: 6.169310092926025s
```

```
INFO:tensorflow:Assets written to:
../reports/models/nn_T_pca_enclosed_office_model.pkl/assets
Mean squared error: 0.2845354445140766
```



[16]:
```
# Save training times to a csv file
import csv

with open('../reports/data/training_times_enclosed_office_T.csv', 'w') as␣
↪csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Model', 'Training time (s)'])
    writer.writerow(['Linear regression', lr_train_time])
    writer.writerow(['Linear regression with PCA', lr_pca_train_time])
    writer.writerow(['Neural network', nn_train_time])
    writer.writerow(['Neural network with PCA', nn_pca_train_time])
```

1.1.5 5. Compare models' results with and without PCA on predicting V

5.1 Split the data into training and testing

[17]:
```
# Split the data into training and testing
from sklearn.model_selection import train_test_split

# Without PCA
X_train, X_test, y_train, y_test = train_test_split(X_std, y_V, test_size=0.2,␣
↪random_state=42)


# With PCA
```

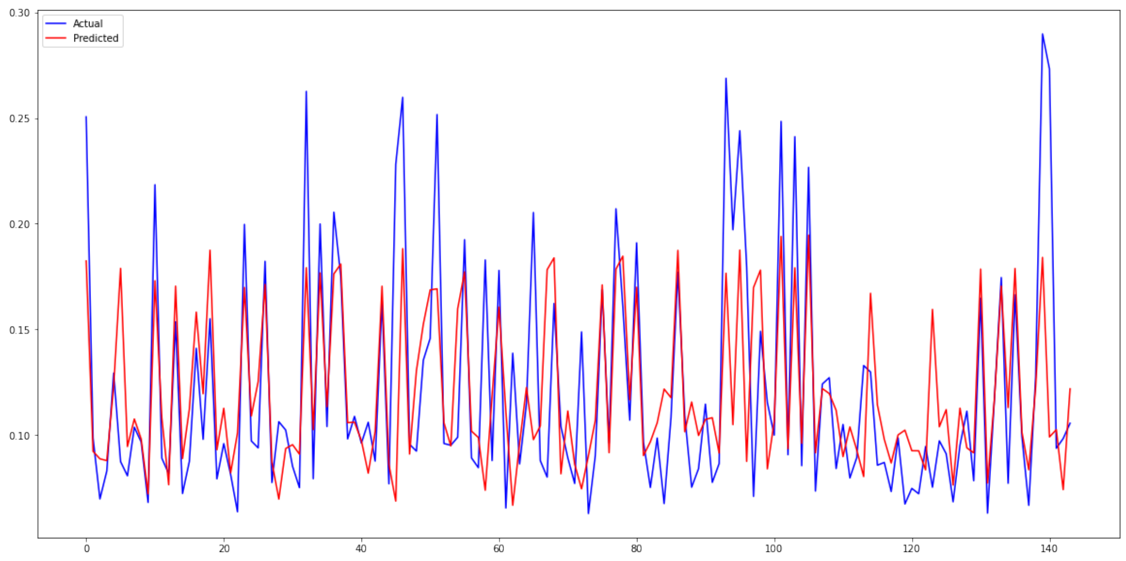
```
X_train_pca, X_test_pca, y_train_pca, y_test_pca = train_test_split(X_pca, y_V,␣
↪test_size=0.2, random_state=42)
```

5.2 Train and evaluate the Linear Regression model

5.2.1 Without PCA

[18]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pickle
import time

# Train linear regression model on training data
start = time.time()
lr_model.fit(X_train, y_train)
stop = time.time()
lr_train_time = stop - start
print(f"Training time: {lr_train_time}s")

# Save the model to a pickle file
filename = '../reports/models/lr_V_enclosed_office_model.pkl'
pickle.dump(lr_model, open(filename, 'wb'))

# Predict on test data
y_pred = lr_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

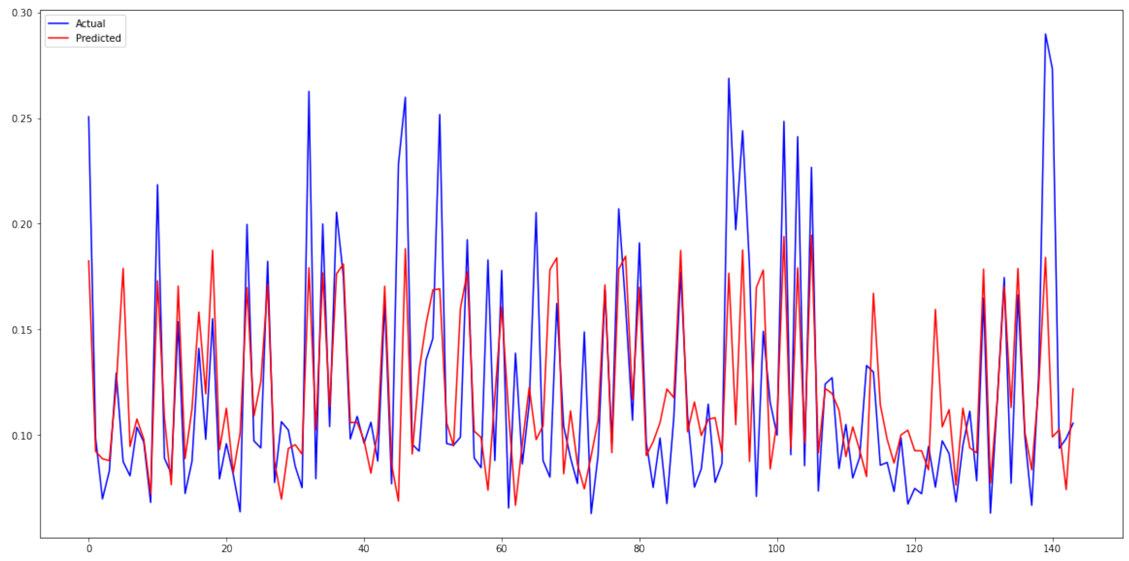
```
Training time: 0.004587411880493164s
Mean squared error: 0.0017227128759820663
```



5.2.2 With PCA

[19]:
```python
# Train linear regression model on training data
start = time.time()
lr_model.fit(X_train_pca, y_train_pca)
stop = time.time()
lr_pca_train_time = stop - start
print(f"Training time: {lr_pca_train_time}s")

# Predict on test data
y_pred_pca = lr_model.predict(X_test_pca)

# Save the model to a pickle file
filename = '../reports/models/lr_V_pca_enclosed_office_model.pkl'
pickle.dump(lr_model, open(filename, 'wb'))

# Evaluate the model
mse = mean_squared_error(y_test_pca, y_pred_pca)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test_pca[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred_pca[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

```
Training time: 0.006722688674926758s
Mean squared error: 0.0017227128759820663
```

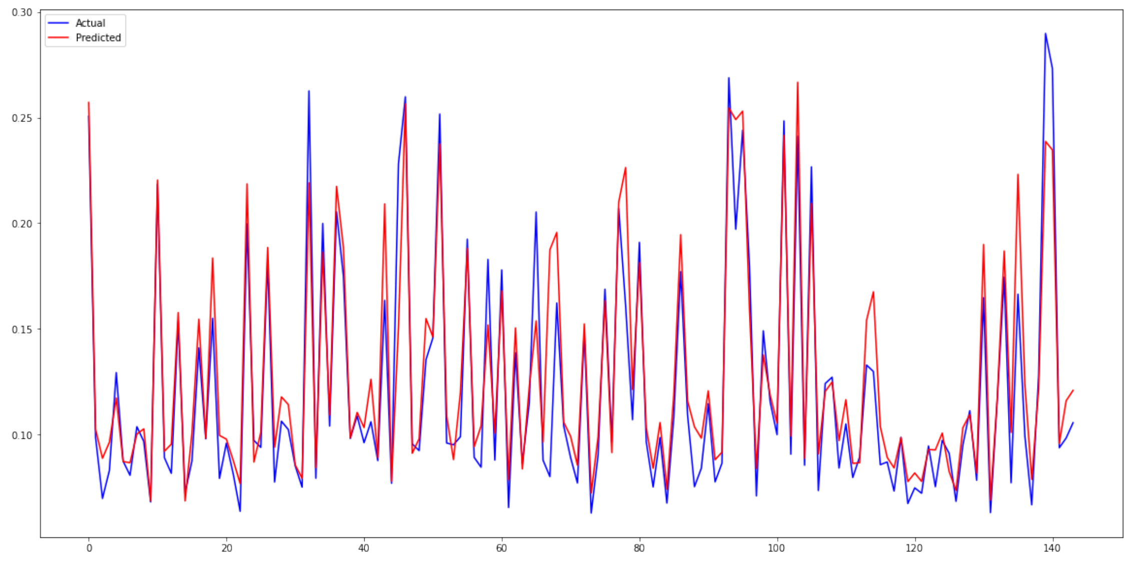


5.3 Train and evaluate the Feedforward Neural Network model

5.2.1 Without PCA

[20]:
```python
# Create neural network model
from keras.models import Sequential
from keras.layers import Dense

# Train the model
start = time.time()
nn_model.fit(X_train, y_train, epochs=100, verbose=0)
stop = time.time()
nn_train_time = stop - start
print(f"Training time: {nn_train_time}s")

# Predict on test data
y_pred_nn = nn_model.predict(X_test)
nn_model.save('../reports/models/nn_V_enclosed_office_model.pkl')

# Evaluate the model
mse = mean_squared_error(y_test, y_pred_nn)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred_nn[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

```
Training time: 6.210849046707153s
INFO:tensorflow:Assets written to:
../reports/models/nn_V_enclosed_office_model.pkl/assets
Mean squared error: 0.0004093002488979166
```

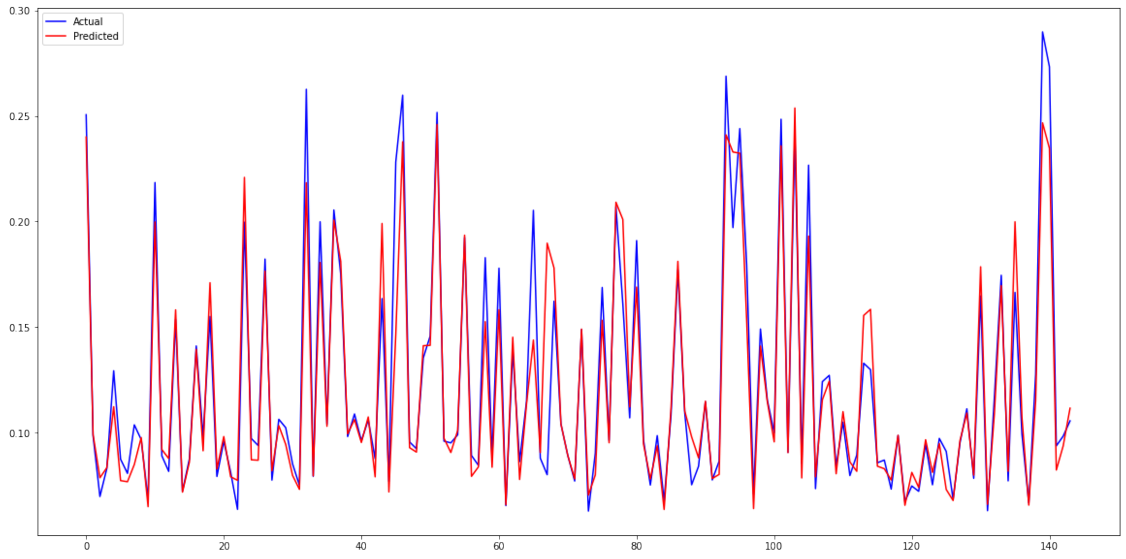


5.3.2 With PCA

[21]:
```python
# Train the model
start = time.time()
nn_model.fit(X_train_pca, y_train_pca, epochs=100, verbose=0)
stop = time.time()
nn_pca_train_time = stop - start
print(f"Training time: {nn_pca_train_time}s")

# Predict on test data
y_pred_nn_pca = nn_model.predict(X_test_pca)
nn_model.save('../reports/models/nn_V_pca_enclosed_office_model.pkl')

# Evaluate the model
mse = mean_squared_error(y_test_pca, y_pred_nn_pca)
print(f'Mean squared error: {mse}')

# Plot the predictions and actual values
plt.figure(figsize=(20,10))
plt.plot(range(144), y_test_pca[:144], color='blue', label='Actual')
plt.plot(range(144), y_pred_nn_pca[:144], color='red', label='Predicted')
plt.legend()
plt.show()
```

```
Training time: 6.2445244789123535s
```

```
INFO:tensorflow:Assets written to:
../reports/models/nn_V_pca_enclosed_office_model.pkl/assets
Mean squared error: 0.00032082510381555888
```



[22]:
```
# Save training times to a csv file
import csv

with open('../reports/data/training_times_enclosed_office_V.csv', 'w') as␣
↪csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Model', 'Training time (s)'])
    writer.writerow(['Linear regression', lr_train_time])
    writer.writerow(['Linear regression with PCA', lr_pca_train_time])
    writer.writerow(['Neural network', nn_train_time])
    writer.writerow(['Neural network with PCA', nn_pca_train_time])
```