TimeSeries_KnappStephen_FinalCode

May 19, 2020

1 Save Figures Function

[9]:
```
import os

# Where to save the figures
PROJECT_ROOT_DIR = "."
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images")
os.makedirs(IMAGES_PATH, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        pyplot.tight_layout()
    pyplot.savefig(path, format=fig_extension, dpi=resolution)
```

2 Load and Prepare the Data

[10]:
```
# load and clean-up power usage data
from numpy import nan
from pandas import read_csv
# load all data
dataset = read_csv('household_power_consumption.txt', sep=';', header=0,␣
↪low_memory=False, infer_datetime_format=True, parse_dates={'datetime':
↪[0,1]}, index_col=['datetime'])
# summarize
print(dataset.shape)
print(dataset.head())
# mark all missing values
dataset.replace('?', nan, inplace=True)
# add a column for the remainder of sub metering
values = dataset.values.astype('float32')
dataset['sub_metering_4'] = (values[:,0] * 1000 / 60) - (values[:,4] + values[:
↪,5] + values[:,6])
# save updated dataset
dataset.to_csv('household_power_consumption.csv')
```

```python
# load the new dataset and summarize
dataset = read_csv('household_power_consumption.csv', header=0,␣
↪infer_datetime_format=True, parse_dates=['datetime'], index_col=['datetime'])
print(dataset.head())
```

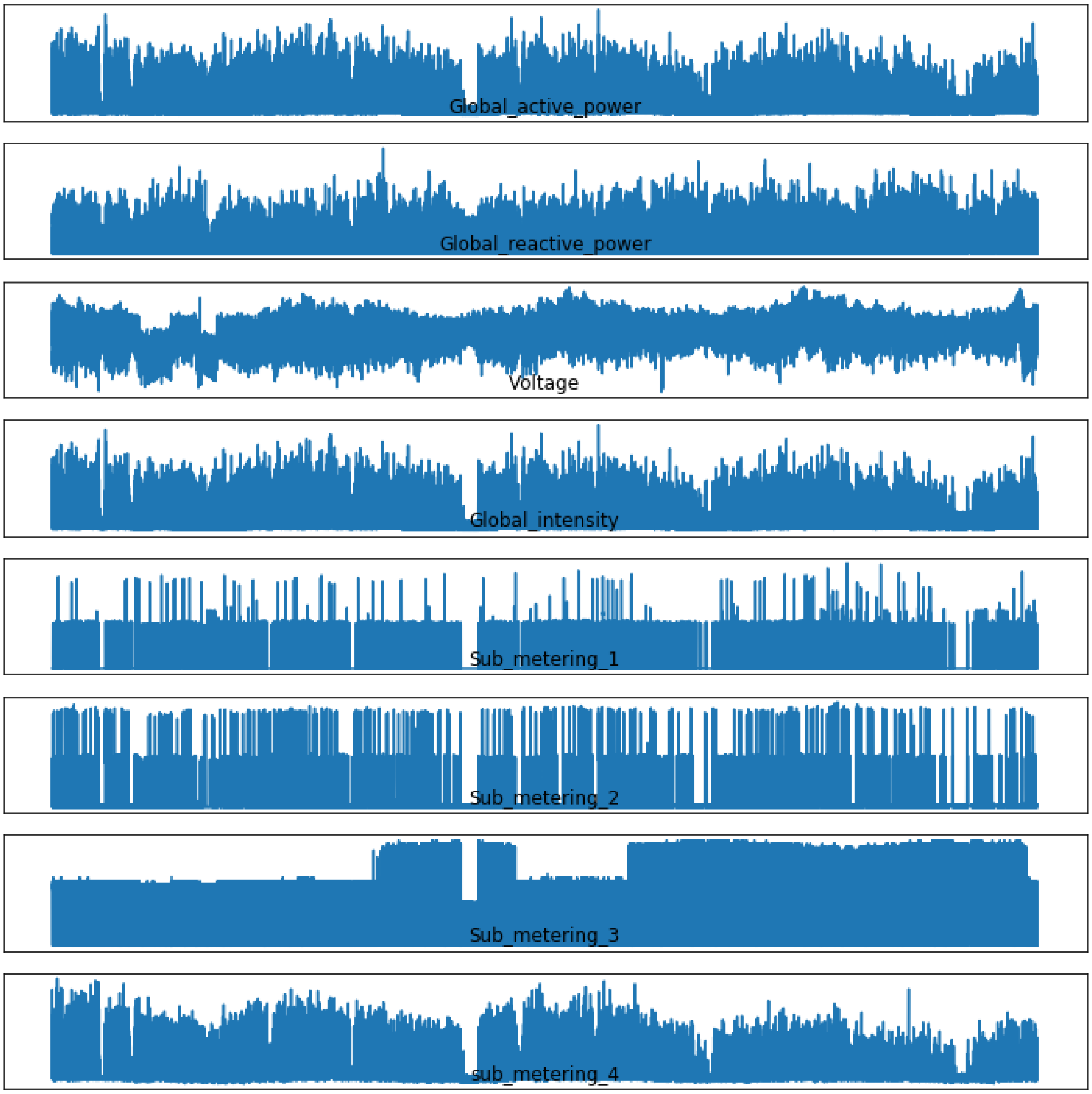
```
(2075259, 7)
                    Global_active_power Global_reactive_power  Voltage  \
datetime
2006-12-16 17:24:00               4.216                 0.418  234.840
2006-12-16 17:25:00               5.360                 0.436  233.630
2006-12-16 17:26:00               5.374                 0.498  233.290
2006-12-16 17:27:00               5.388                 0.502  233.740
2006-12-16 17:28:00               3.666                 0.528  235.680

                    Global_intensity Sub_metering_1 Sub_metering_2  \
datetime
2006-12-16 17:24:00           18.400          0.000          1.000
2006-12-16 17:25:00           23.000          0.000          1.000
2006-12-16 17:26:00           23.000          0.000          2.000
2006-12-16 17:27:00           23.000          0.000          1.000
2006-12-16 17:28:00           15.800          0.000          1.000

                     Sub_metering_3
datetime
2006-12-16 17:24:00            17.0
2006-12-16 17:25:00            16.0
2006-12-16 17:26:00            17.0
2006-12-16 17:27:00            17.0
2006-12-16 17:28:00            17.0
                     Global_active_power  Global_reactive_power  Voltage  \
datetime
2006-12-16 17:24:00                4.216                  0.418   234.84
2006-12-16 17:25:00                5.360                  0.436   233.63
2006-12-16 17:26:00                5.374                  0.498   233.29
2006-12-16 17:27:00                5.388                  0.502   233.74
2006-12-16 17:28:00                3.666                  0.528   235.68

                     Global_intensity  Sub_metering_1  Sub_metering_2  \
datetime
2006-12-16 17:24:00              18.4             0.0             1.0
2006-12-16 17:25:00              23.0             0.0             1.0
2006-12-16 17:26:00              23.0             0.0             2.0
2006-12-16 17:27:00              23.0             0.0             1.0
2006-12-16 17:28:00              15.8             0.0             1.0

                     Sub_metering_3  sub_metering_4
datetime
```

```
2006-12-16 17:24:00            17.0       52.266670
2006-12-16 17:25:00            16.0       72.333336
2006-12-16 17:26:00            17.0       70.566666
2006-12-16 17:27:00            17.0       71.800000
2006-12-16 17:28:00            17.0       43.100000
```

3 Explore the Data

[11]:
```python
# line plots for power usage dataset
from pandas import read_csv
from matplotlib import pyplot
# line plot for each variable
pyplot.figure(figsize=(10,10))
for i in range(len(dataset.columns)):
        # create subplot
        pyplot.subplot(len(dataset.columns), 1, i+1)
        # get variable name
        name = dataset.columns[i]
        # plot data
        pyplot.plot(dataset[name])
        # set title
        pyplot.title(name, y=0)
        # turn off ticks to remove clutter
        pyplot.yticks([])
        pyplot.xticks([])
save_fig("exploratory plots1")
pyplot.show()
```

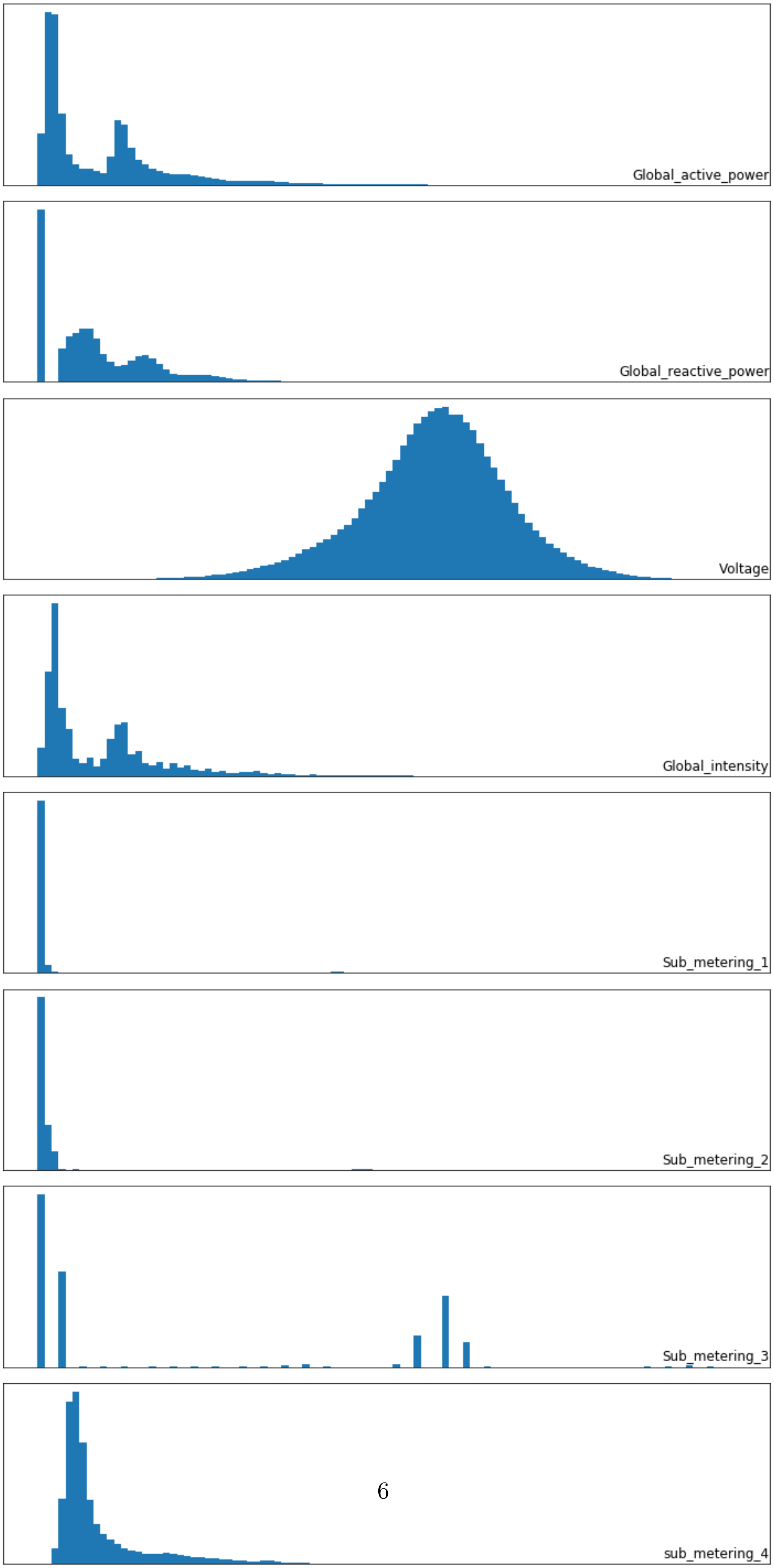
```
Saving figure exploratory plots1
```



[12]:
```
# histogram plots for power usage dataset
from pandas import read_csv
from matplotlib import pyplot
# histogram plot for each variable
pyplot.figure(figsize=(10,20))
for i in range(len(dataset.columns)):
        # create subplot
        pyplot.subplot(len(dataset.columns), 1, i+1)
        # get variable name
        name = dataset.columns[i]
        # create histogram
        dataset[name].hist(bins=100)
        # set title
```

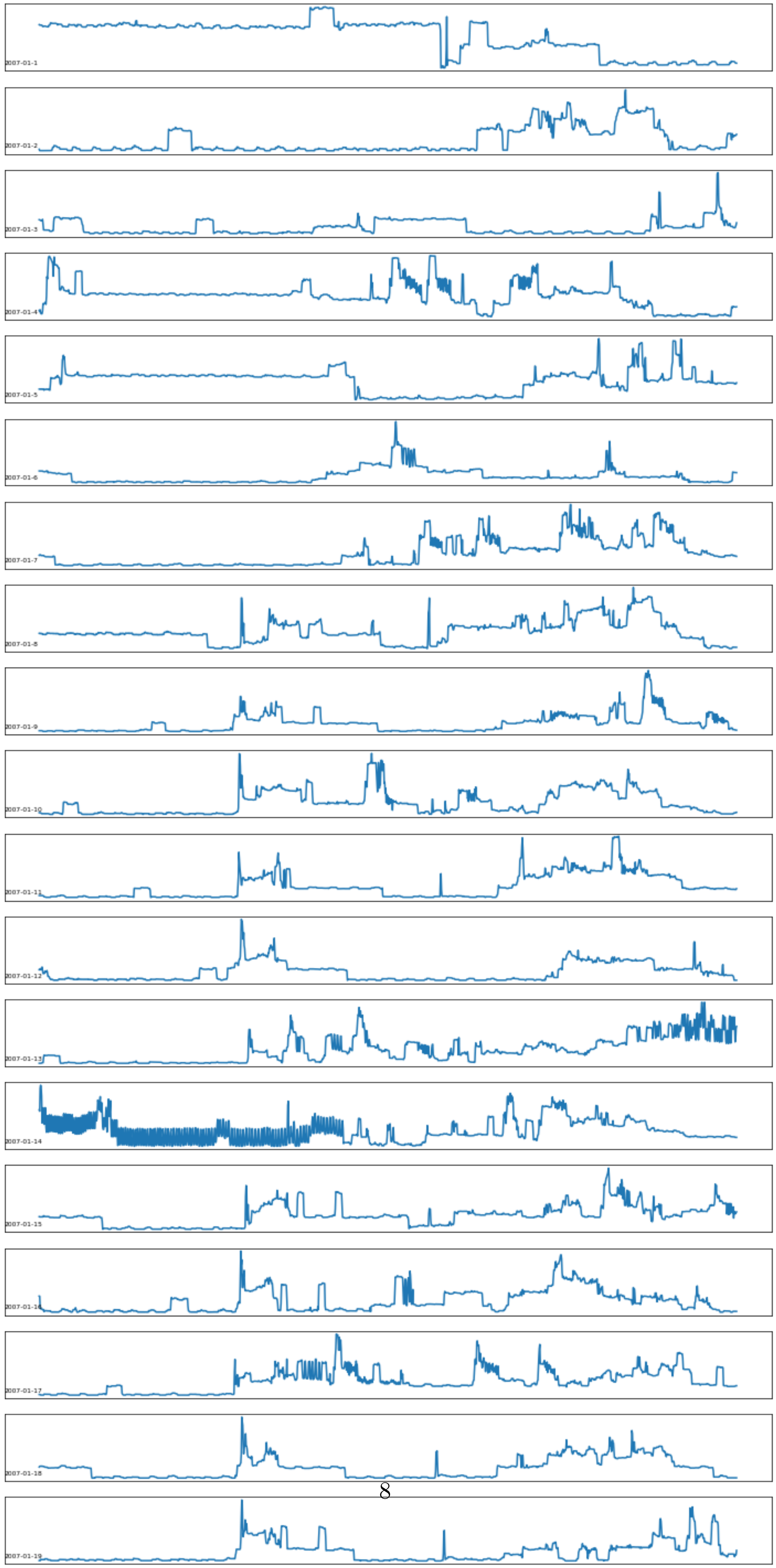
```
        pyplot.title(name, y=0, loc='right')
        # turn off ticks to remove clutter
        pyplot.yticks([])
        pyplot.xticks([])
save_fig("exploratory plots2")
pyplot.show()
```

```
Saving figure exploratory plots2
```


Global_active_power
Global_reactive_power
Voltage
Global_intensity
Sub_metering_1
Sub_metering_2
Sub_metering_3
sub_metering_4

[13]:
```python
# daily line plots for power usage dataset
from pandas import read_csv
from matplotlib import pyplot
# plot active power for each year
days = [x for x in range(1, 20)]
pyplot.figure(figsize=(10,20))
for i in range(len(days)):
        # prepare subplot
        ax = pyplot.subplot(len(days), 1, i+1)
        # determine the day to plot
        day = '2007-01-' + str(days[i])
        # get all observations for the day
        result = dataset[day]
        # plot the active power for the day
        pyplot.plot(result['Global_active_power'])
        # add a title to the subplot
        pyplot.title(day, y=0, loc='left', size=6)
        # turn off ticks to remove clutter
        pyplot.yticks([])
        pyplot.xticks([])
save_fig("exploratory plots3")
pyplot.show()
```

```
Saving figure exploratory plots3
```


2007-01-1
2007-01-2
2007-01-3
2007-01-4
2007-01-5
2007-01-6
2007-01-7
2007-01-8
2007-01-9
2007-01-10
2007-01-11
2007-01-12
2007-01-13
2007-01-14
2007-01-15
2007-01-16
2007-01-17
2007-01-18
2007-01-19

3.1 Convert Data to Daily

[14]:
```python
# resample minute data to total for each day for the power usage dataset
from pandas import read_csv
# load the new file
dataset = read_csv('household_power_consumption.csv', header=0,␣
 ↪infer_datetime_format=True, parse_dates=['datetime'], index_col=['datetime'])
# resample data to daily
daily_groups = dataset.resample('D')
daily_data = daily_groups.sum()
# summarize
print(daily_data.shape)
print(daily_data.head())
# save
daily_data.to_csv('household_power_consumption_days.csv')
```

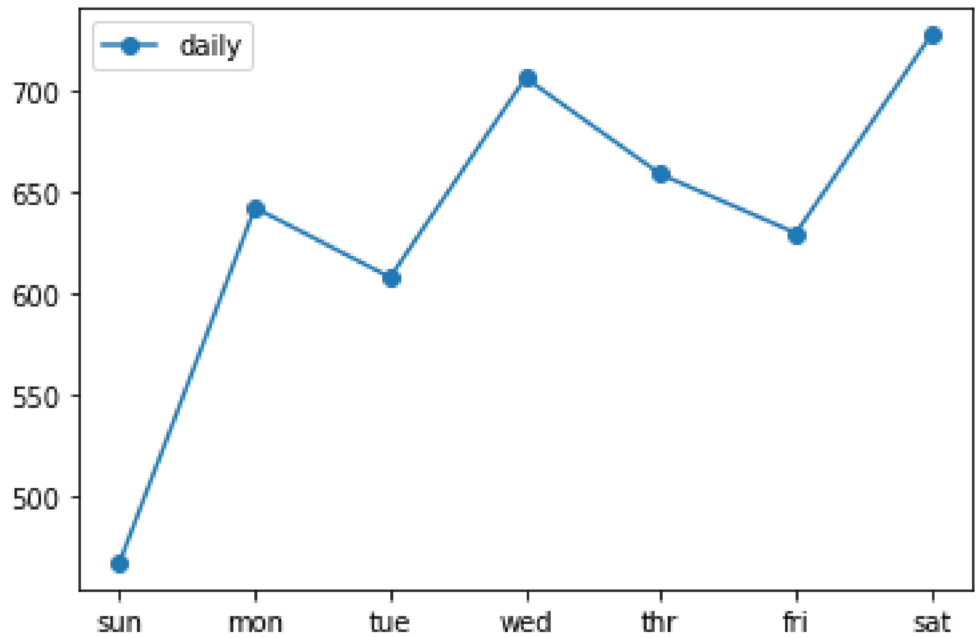
```
(1442, 8)
            Global_active_power  Global_reactive_power    Voltage  \
datetime
2006-12-16             1209.176                 34.922   93552.53
2006-12-17             3390.460                226.006  345725.32
2006-12-18             2203.826                161.792  347373.64
2006-12-19             1666.194                150.942  348479.01
2006-12-20             2225.748                160.998  348923.61

            Global_intensity  Sub_metering_1  Sub_metering_2  Sub_metering_3  \
datetime
2006-12-16            5180.8             0.0           546.0          4926.0
2006-12-17           14398.6          2033.0          4187.0         13341.0
2006-12-18            9247.2          1063.0          2621.0         14018.0
2006-12-19            7094.0           839.0          7602.0          6197.0
2006-12-20            9313.0             0.0          2648.0         14063.0

            sub_metering_4
datetime
2006-12-16    14680.933319
2006-12-17    36946.666732
2006-12-18    19028.433281
2006-12-19    13131.900043
2006-12-20    20384.800011
```

4 Baseline Naive Model

[16]:
```
# naive forecast strategies for the power usage dataset
import pandas as pd
from math import sqrt
from numpy import split
from numpy import array
from pandas import read_csv
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot

# split a univariate dataset into train/test sets
def split_dataset(data):
        # split into standard weeks
        train, test = data[1:-328], data[-328:-6]
        # restructure into windows of weekly data
        train = array(split(train, len(train)/7))
        test = array(split(test, len(test)/7))
        return train, test

# evaluate one or more weekly forecasts against expected values
def evaluate_forecasts(actual, predicted):
        scores = list()
        # calculate an RMSE score for each day
        for i in range(actual.shape[1]):
                # calculate mse
                mse = mean_squared_error(actual[:, i], predicted[:, i])
                # calculate rmse
                rmse = sqrt(mse)
                # store
                scores.append(rmse)
        # calculate overall RMSE
        s = 0
        for row in range(actual.shape[0]):
                for col in range(actual.shape[1]):
                        s += (actual[row, col] - predicted[row, col])**2
        score = sqrt(s / (actual.shape[0] * actual.shape[1]))
        return score, scores

# summarize scores
def summarize_scores(name, score, scores):
        s_scores = ', '.join(['%.1f' % s for s in scores])
        print('%s: [%.3f] %s' % (name, score, s_scores))

# evaluate a single model
def evaluate_model(model_func, train, test):
        # history is a list of weekly data
```

```
        history = [x for x in train]
        # walk-forward validation over each week
        predictions = list()
        for i in range(len(test)):
                # predict the week
                yhat_sequence = model_func(history)
                # store the predictions
                predictions.append(yhat_sequence)
                # get real observation and add to history for predicting the␣
 ↪next week
                history.append(test[i, :])
        predictions = array(predictions)
        # evaluate predictions days for each week
        score, scores = evaluate_forecasts(test[:, :, 0], predictions)
        return score, scores, predictions

# daily persistence model
def daily_persistence(history):
        # get the data for the prior week
        last_week = history[-1]
        # get the total active power for the last day
        value = last_week[-1, 0]
        # prepare 7 day forecast
        forecast = [value for _ in range(7)]
        return forecast

# load the new file
dataset = read_csv('household_power_consumption_days.csv', header=0,␣
 ↪infer_datetime_format=True, parse_dates=['datetime'], index_col=['datetime'])
# split into train and test
train, test = split_dataset(dataset.values)
# define the names and functions for the models we wish to evaluate
models = dict()
models['daily'] = daily_persistence
# evaluate each model
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
for name, func in models.items():
        # evaluate and get scores
        score, scores, predictions = evaluate_model(func, train, test)
        # summarize scores
        summarize_scores(name, score, scores)
        # plot scores
        pyplot.plot(days, scores, marker='o', label=name)
# show plot
pyplot.legend()
pyplot.show()
```

```
daily: [638.933] 467.4, 642.0, 607.9, 706.1, 658.6, 629.6, 727.2
```



5 Multi-step encoder-decoder LSTM

5.1 Common Functions

[17]:
```
# multivariate multi-step encoder-decoder lstm for the power usage dataset
from math import sqrt
from numpy import split
from numpy import array
import pandas as pd
from pandas import read_csv
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import RepeatVector
from tensorflow.keras.layers import TimeDistributed

# split a univariate dataset into train/test sets
def split_dataset(data):
        # split into standard weeks
        train, test = data[1:-328], data[-328:-6]
        # restructure into windows of weekly data
```

```
	train = array(split(train, len(train)/7))
	test = array(split(test, len(test)/7))
	return train, test

# evaluate one or more weekly forecasts against expected values
def evaluate_forecasts(actual, predicted):
	scores = list()
	# calculate an RMSE score for each day
	for i in range(actual.shape[1]):
		# calculate mse
		mse = mean_squared_error(actual[:, i], predicted[:, i])
		# calculate rmse
		rmse = sqrt(mse)
		# store
		scores.append(rmse)
	# calculate overall RMSE
	s = 0
	for row in range(actual.shape[0]):
		for col in range(actual.shape[1]):
			s += (actual[row, col] - predicted[row, col])**2
	score = sqrt(s / (actual.shape[0] * actual.shape[1]))
	return score, scores

# summarize scores
def summarize_scores(name, score, scores):
	s_scores = ', '.join(['%.1f' % s for s in scores])
	print('%s: [%.3f] %s' % (name, score, s_scores))

# convert history into inputs and outputs
def to_supervised(train, n_input, n_out=7):
	# flatten data
	data = train.reshape((train.shape[0]*train.shape[1], train.shape[2]))
	X, y = list(), list()
	in_start = 0
	# step over the entire history one time step at a time
	for _ in range(len(data)):
		# define the end of the input sequence
		in_end = in_start + n_input
		out_end = in_end + n_out
		# ensure we have enough data for this instance
		if out_end <= len(data):
			X.append(data[in_start:in_end, :])
			y.append(data[in_end:out_end, 0])
		# move along one time step
		in_start += 1
	return array(X), array(y)
```

```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 50, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
↪n_features)))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
↪verbose=verbose)
        return model

# make a forecast
def forecast(model, history, n_input):
        # flatten data
        data = array(history)
        data = data.reshape((data.shape[0]*data.shape[1], data.shape[2]))
        # retrieve last observations for input data
        input_x = data[-n_input:, :]
        # reshape into [1, n_input, n]
        input_x = input_x.reshape((1, input_x.shape[0], input_x.shape[1]))
        # forecast the next week
        yhat = model.predict(input_x, verbose=0)
        # we only want the vector forecast
        yhat = yhat[0]
        return yhat

# evaluate a single model
def evaluate_model(train, test, n_input):
        # fit model
        model = build_model(train, n_input)
        # history is a list of weekly data
        history = [x for x in train]
        # walk-forward validation over each week
        predictions = list()
```

```
        for i in range(len(test)):
                # predict the week
                yhat_sequence = forecast(model, history, n_input)
                # store the predictions
                predictions.append(yhat_sequence)
                # get real observation and add to history for predicting the␣
 ↪next week
                history.append(test[i, :])
        # evaluate predictions days for each week
        predictions = array(predictions)
        score, scores = evaluate_forecasts(test[:, :, 0], predictions)
        return score, scores, predictions

# load the new file
dataset = read_csv('household_power_consumption_days.csv', header=0,␣
 ↪infer_datetime_format=True, parse_dates=['datetime'], index_col=['datetime'])
# split into train and test
train, test = split_dataset(dataset.values)
# aggregate test data for comparison later
test_df = pd.DataFrame.from_records(test)
test_values = test_df.mean(axis=0)
# number of days to train on
n_input = 28
```

5.2 Base Model: 200 neurons, 50 epochs, 2e/2d, dropout=0, regularization=none

[18]:

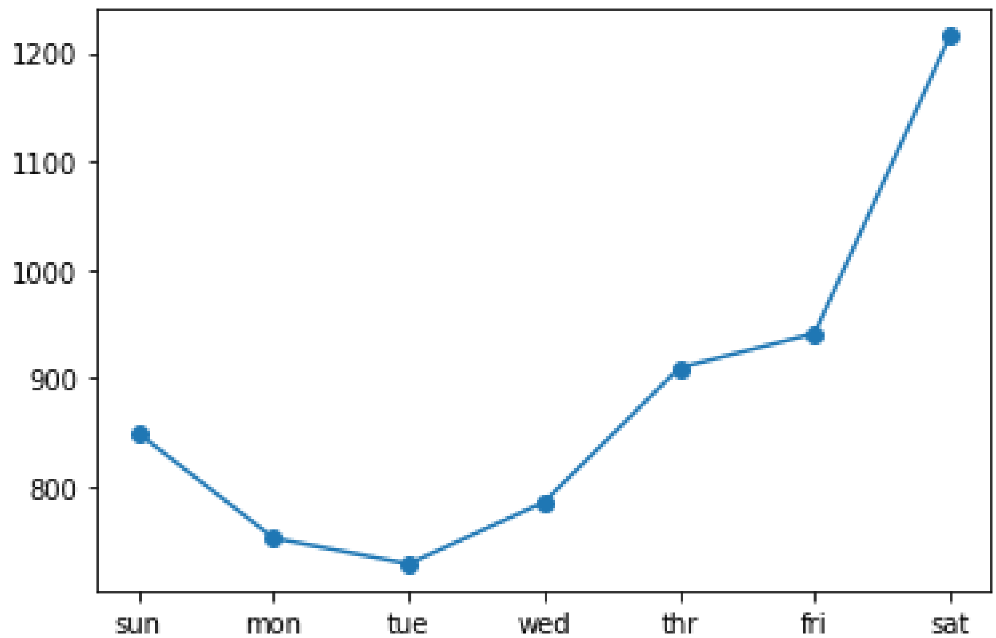
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 50, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
 ↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
 ↪n_features)))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
```

```
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
↪verbose=verbose)
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("BaseModel")
pyplot.show()
```

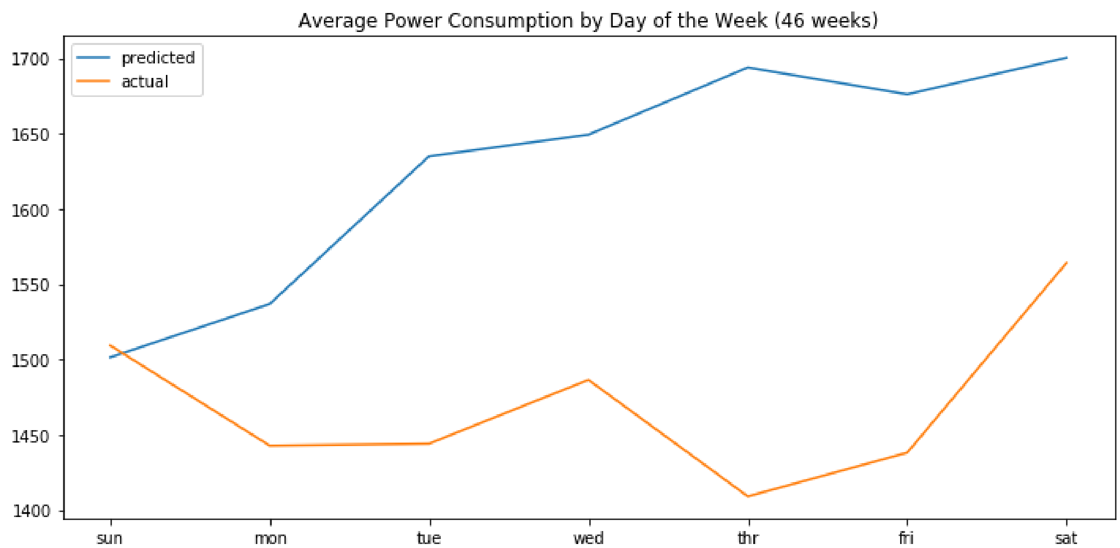
```
Train on 1079 samples
Epoch 1/50
1079/1079 [==============================] - 5s 4ms/sample - loss:
826724439.5255
Epoch 2/50
1079/1079 [==============================] - 2s 2ms/sample - loss:
263395121.9425
Epoch 3/50
1079/1079 [==============================] - 2s 2ms/sample - loss:
202326330.1983
Epoch 4/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 41889770.6172
Epoch 5/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 3074295.9106
Epoch 6/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 3235055.7883
Epoch 7/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1487544.0449
Epoch 8/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 843395.3422
Epoch 9/50
```

```
1079/1079 [==============================] - 2s 2ms/sample - loss: 736880.6382
Epoch 10/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 5493450.4032
Epoch 11/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 49146213.8424
Epoch 12/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 8706023.4458
Epoch 13/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 5571188.1701
Epoch 14/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 40586589.4419
Epoch 15/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1129941.9409
Epoch 16/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 836932.9945
Epoch 17/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 844289.8723
Epoch 18/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1032687.5133
Epoch 19/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 647557.6032
Epoch 20/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 578393.6492
Epoch 21/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 498944.9239
Epoch 22/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 459508.0966
Epoch 23/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 455510.7981
Epoch 24/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 898464.5625
Epoch 25/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 617396.6026
Epoch 26/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 443059.7788
Epoch 27/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 513709.6903
Epoch 28/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 447711.5253
Epoch 29/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 461187.4777
Epoch 30/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 411807.7528
Epoch 31/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 409242.9210
Epoch 32/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 375678.1841
Epoch 33/50
```

```
1079/1079 [==============================] - 2s 2ms/sample - loss: 378075.6036
Epoch 34/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 706195.7030
Epoch 35/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 409733.8848
Epoch 36/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 408680.0287
Epoch 37/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 453228.0697
Epoch 38/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 438857.8571
Epoch 39/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 400921.3695
Epoch 40/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 476260.3896
Epoch 41/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 400376.1967
Epoch 42/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 373920.7715
Epoch 43/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 390953.9657
Epoch 44/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1276909.4087
Epoch 45/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 3357219.7201
Epoch 46/50
1079/1079 [==============================] - 2s 2ms/sample - loss:
113935929.3021
Epoch 47/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 2544199.2964
Epoch 48/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 652011.2124
Epoch 49/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 671933.2324
Epoch 50/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 440009.4317
lstm: [896.501] 849.3, 752.4, 728.5, 785.7, 909.3, 941.1, 1215.6
```




```
Saving figure BaseModel
```



[19]:
```
scores
```

[19]:
```
[849.3424868466016,
 752.4247649356417,
 728.513763472444,
```

```
785.71412402286,
909.3320157483793,
941.1169218118878,
1215.6498003722531]
```

5.3 Epoch Experiment

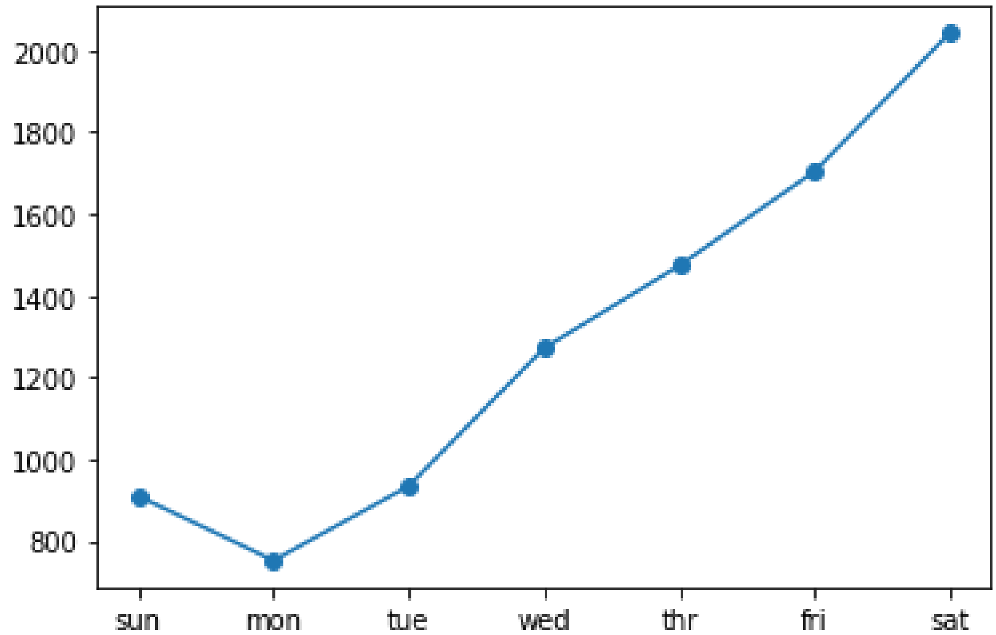
5.3.1 200 neurons, 20 epochs, 2e/2d, dropout=0, regularization=none

[20]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 20, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
↪n_features)))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
↪verbose=verbose)
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
```

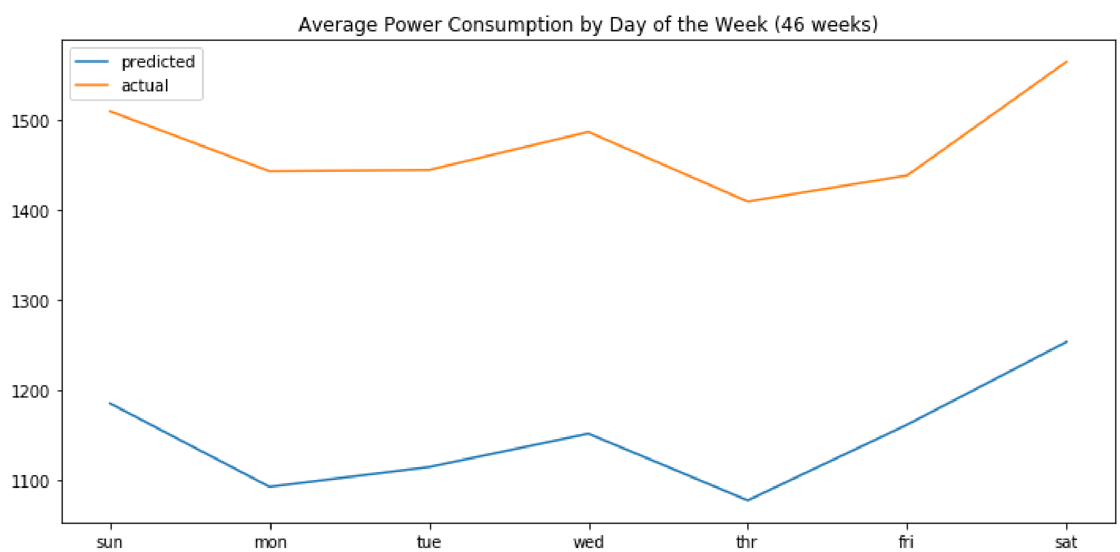
```
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("Epochs20")
pyplot.show()
```

```
Train on 1079 samples
Epoch 1/20
1079/1079 [==============================] - 3s 3ms/sample - loss:
436662910.3540
Epoch 2/20
1079/1079 [==============================] - 2s 2ms/sample - loss:
211727847.2289
Epoch 3/20
1079/1079 [==============================] - 2s 2ms/sample - loss:
109559645.3457
Epoch 4/20
1079/1079 [==============================] - 2s 2ms/sample - loss:
204214269.1010
Epoch 5/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 3439626.8228
Epoch 6/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 608407.8133
Epoch 7/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 564816.7854
Epoch 8/20
1079/1079 [==============================] - ETA: 0s - loss: 505704.26 - 2s
2ms/sample - loss: 503455.7256
Epoch 9/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 456920.5689
Epoch 10/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 449394.7853
Epoch 11/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 422344.4574
Epoch 12/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 1511953.3760
Epoch 13/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 1205332.0553
Epoch 14/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 1970785.9232
Epoch 15/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 844267.1386
Epoch 16/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 540555.4242
Epoch 17/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 506643.2181
```

```
Epoch 18/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 467317.7678
Epoch 19/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 467059.1126
Epoch 20/20
1079/1079 [==============================] - 2s 2ms/sample - loss: 374686.2011
lstm: [1369.551] 909.9, 752.1, 934.2, 1271.8, 1473.7, 1703.6, 2042.5
```




```
Saving figure Epochs20
```



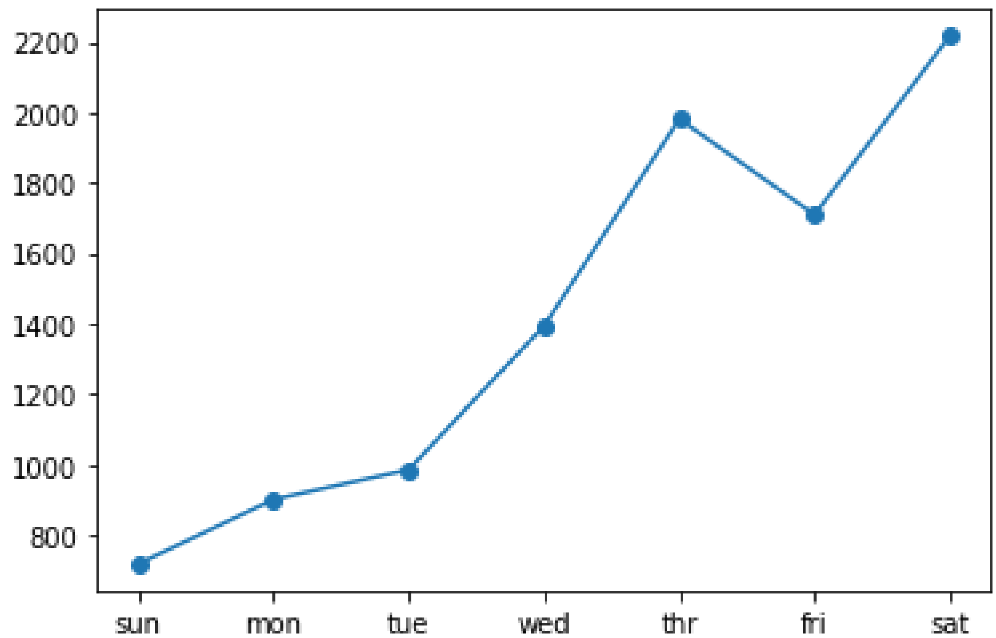
5.3.2 200 neurons, 35 epochs, 2e/2d, dropout=0, regularization=none

[21]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 35, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
 ↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
 ↪n_features)))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
 ↪verbose=verbose)
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("Epochs35")
```

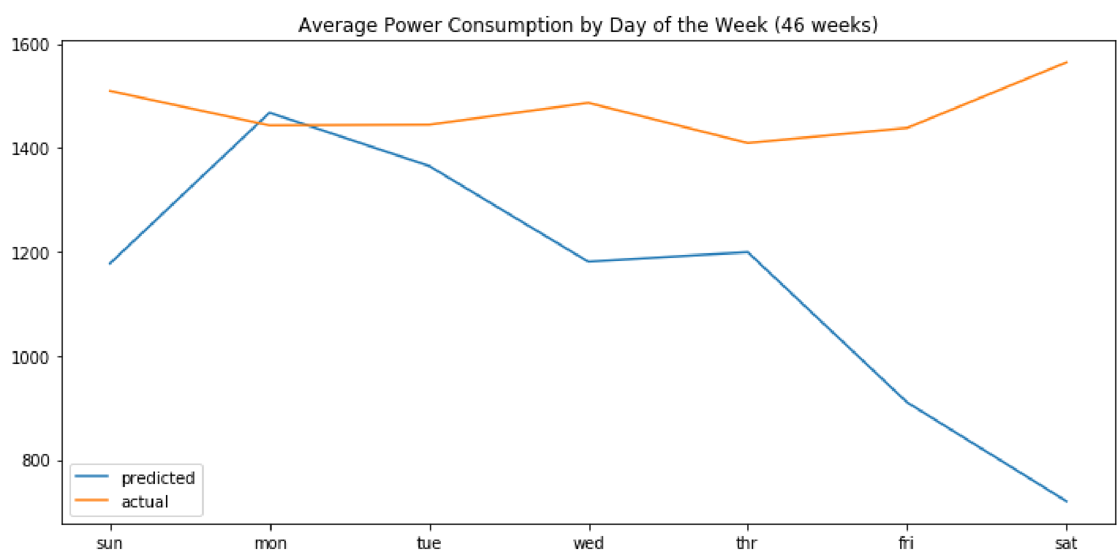
```
pyplot.show()
```

```
Train on 1079 samples
Epoch 1/35
1079/1079 [==============================] - 3s 3ms/sample - loss:
1761650856.4523
Epoch 2/35
1079/1079 [==============================] - 2s 2ms/sample - loss:
320757600.7266
Epoch 3/35
1079/1079 [==============================] - 2s 2ms/sample - loss:
111888687.9852
Epoch 4/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 17224512.8489
Epoch 5/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 23119862.3697
Epoch 6/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3048133.8830
Epoch 7/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 22726674.7256
Epoch 8/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1647335.5123
Epoch 9/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2249845.0917
Epoch 10/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1045571.6637
Epoch 11/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1999965.6641
Epoch 12/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2546869.5777
Epoch 13/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2041431.3477
Epoch 14/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2497443.9722
Epoch 15/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1387001.6148
Epoch 16/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 687950.9285
Epoch 17/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 466018.8196
Epoch 18/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 531299.3267
Epoch 19/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 975346.1102
Epoch 20/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1358009.6375
Epoch 21/35
```

```
1079/1079 [==============================] - 2s 2ms/sample - loss: 2941914.9166
Epoch 22/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 676592.3422
Epoch 23/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 665386.9097
Epoch 24/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 410267.4193
Epoch 25/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 358578.1982
Epoch 26/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 336386.1000
Epoch 27/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 385964.7264
Epoch 28/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 413656.1286
Epoch 29/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 330539.9121
Epoch 30/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 326984.2699
Epoch 31/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 402679.8139
Epoch 32/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 369554.4699
Epoch 33/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 332649.2415
Epoch 34/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 377854.9138
Epoch 35/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 374402.5451
lstm: [1512.103] 717.4, 900.9, 985.5, 1393.7, 1981.4, 1709.5, 2217.4
```



Saving figure Epochs35



5.4 Neurons/Layer and Number of Layers Experiment

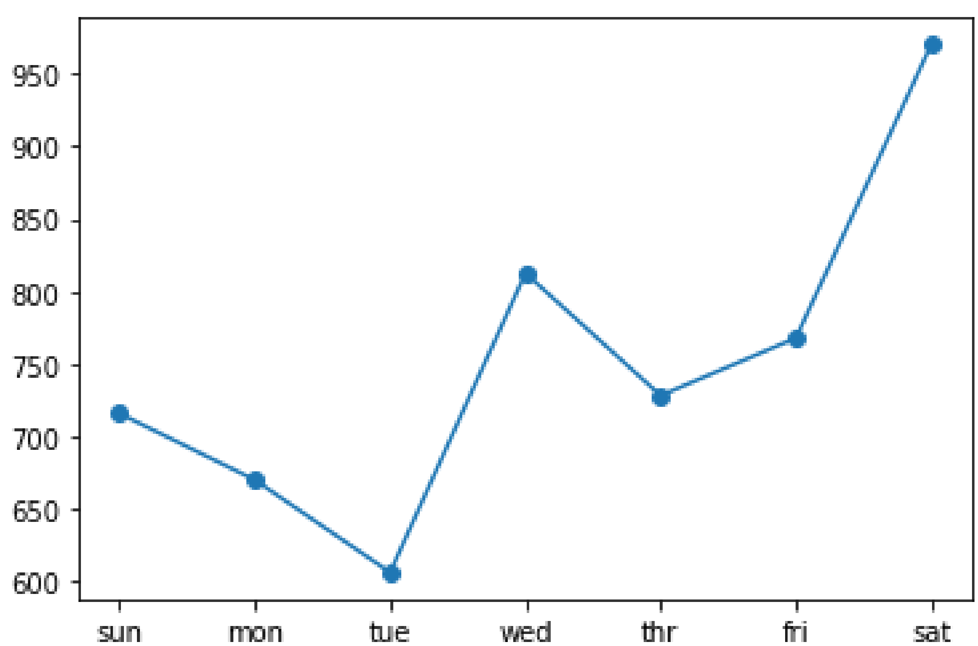
5.4.1 100 neurons, 35 epochs, 2e/2d, dropout=0, regularization=none

[22]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 35, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
 ↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(100, activation='relu', input_shape=(n_timesteps,␣
 ↪n_features)))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(100, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(50, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
 ↪verbose=verbose)
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("Neurons100Decoder2")
pyplot.show()
```

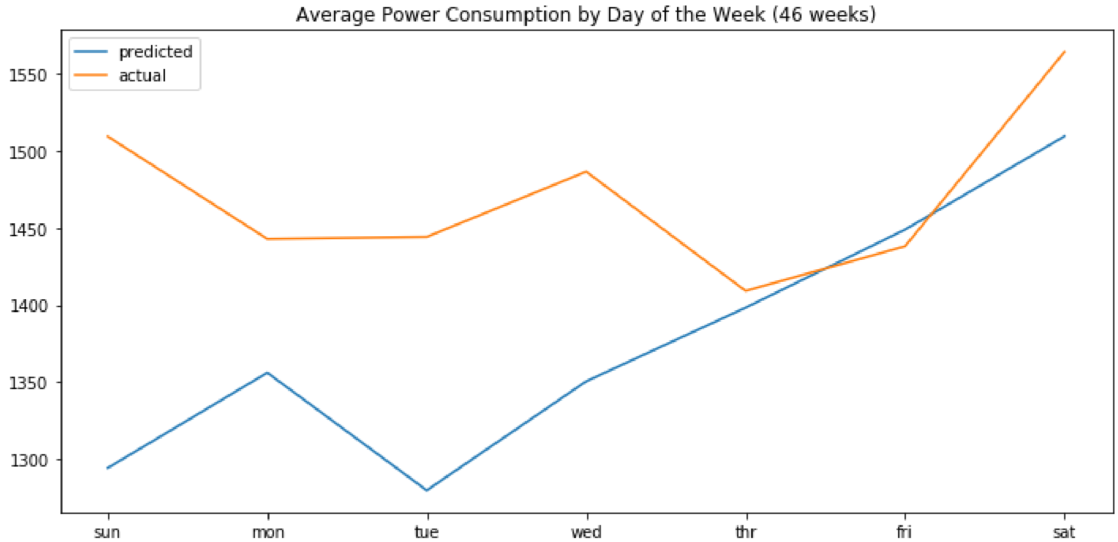
```
Train on 1079 samples
Epoch 1/35
```

```
1079/1079 [==============================] - 2s 2ms/sample - loss:
1627782614.9842
Epoch 2/35
1079/1079 [==============================] - 1s 1ms/sample - loss:
234017739.5737
Epoch 3/35
1079/1079 [==============================] - 1s 1ms/sample - loss:
737185221.9018
Epoch 4/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 91449836.7238
Epoch 5/35
1079/1079 [==============================] - 1s 1ms/sample - loss:
286834062.5839
Epoch 6/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 70397934.9379
Epoch 7/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 25179404.9861
Epoch 8/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 17516880.2954
Epoch 9/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 9827677.7071
Epoch 10/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 5726022.8811
Epoch 11/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 8396755.8184
Epoch 12/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 28105982.3772
Epoch 13/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 3956935.5904
Epoch 14/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 1955337.5589
Epoch 15/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 2033391.3274
Epoch 16/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 1056955.3782
Epoch 17/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 619958.1498
Epoch 18/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 495600.8310
Epoch 19/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 467774.8921
Epoch 20/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 490902.1154
Epoch 21/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 454644.7244
Epoch 22/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 484920.9901
Epoch 23/35
```

```
1079/1079 [==============================] - 1s 1ms/sample - loss: 427613.8303
Epoch 24/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 422705.5261
Epoch 25/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 558554.77530s
- loss: 540
Epoch 26/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 636593.8871
Epoch 27/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 492904.7366
Epoch 28/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 440761.2313
Epoch 29/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 626609.7626
Epoch 30/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 9251150.6617
Epoch 31/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 2250818.5990
Epoch 32/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 950889.5856
Epoch 33/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 451186.3139
Epoch 34/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 627423.7670
Epoch 35/35
1079/1079 [==============================] - 1s 1ms/sample - loss: 558761.8910
lstm: [760.547] 715.6, 669.7, 605.7, 812.6, 727.9, 767.7, 970.6
```



```
Saving figure Neurons100Decoder2
```



5.4.2 400 neurons, 35 epochs, 2e/3d, dropout=0, regularization=none

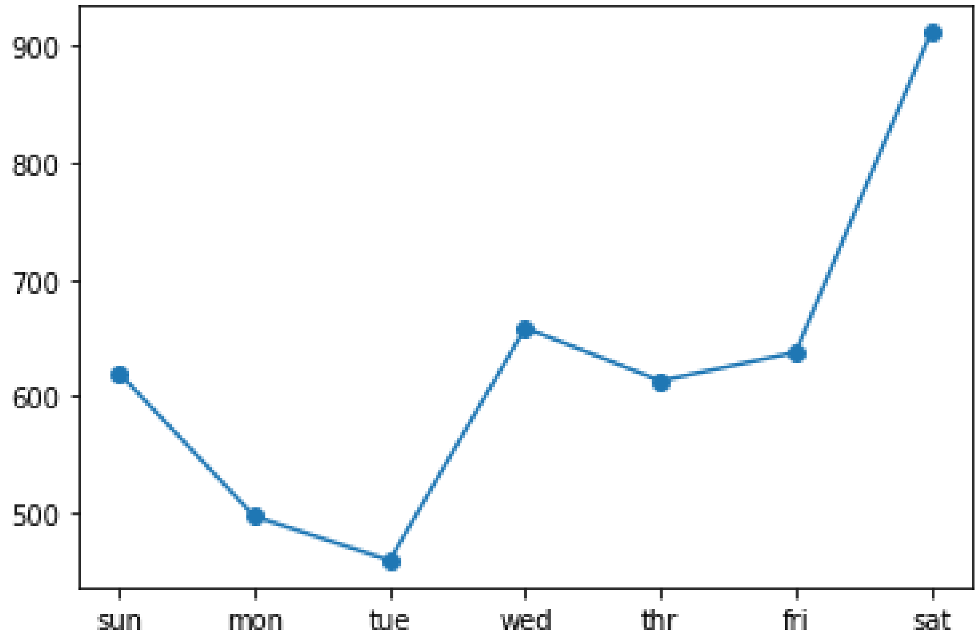
[23]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 35, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
 ↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(400, activation='relu', input_shape=(n_timesteps,␣
 ↪n_features)))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(400, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(200, activation='relu')))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
 ↪verbose=verbose)
```

```
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("Neurons400Decoder3")
pyplot.show()
```

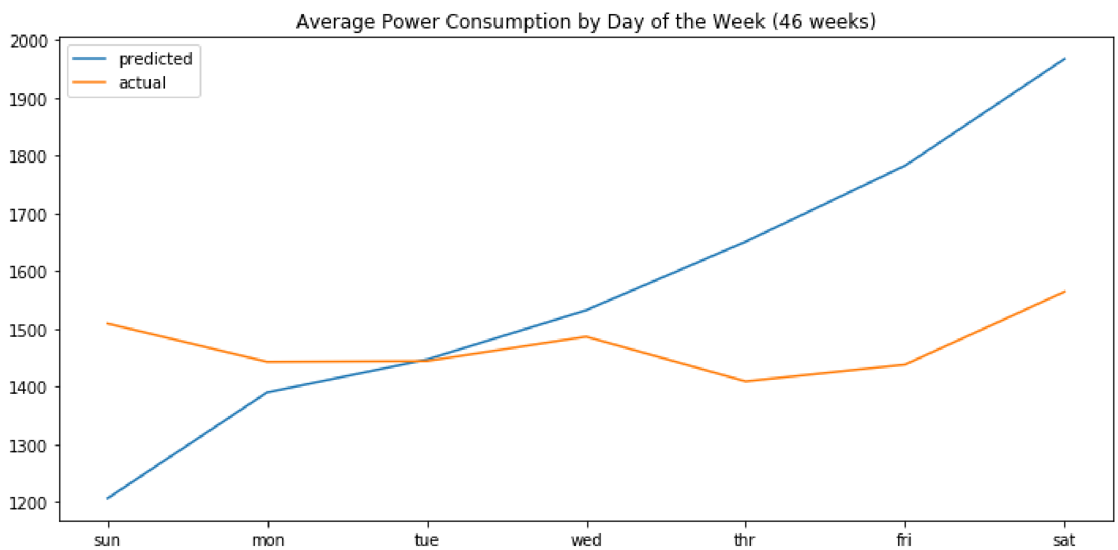
```
Train on 1079 samples
Epoch 1/35
1079/1079 [==============================] - 9s 9ms/sample - loss:
350945652.1816
Epoch 2/35
1079/1079 [==============================] - 8s 7ms/sample - loss:
152675130.0019
Epoch 3/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 51594299.7461
Epoch 4/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 7607225.7257
Epoch 5/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 3793197.6766
Epoch 6/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 6173842.3939
Epoch 7/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 3272474.3698
Epoch 8/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 2533232.3935
Epoch 9/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 893101.7550
Epoch 10/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 526664.6292
Epoch 11/35
```

```
1079/1079 [==============================] - 9s 8ms/sample - loss: 480065.7954
Epoch 12/35
1079/1079 [==============================] - 9s 8ms/sample - loss: 445161.6005
Epoch 13/35
1079/1079 [==============================] - 9s 8ms/sample - loss: 644560.8407
Epoch 14/35
1079/1079 [==============================] - 9s 8ms/sample - loss: 613007.6233
Epoch 15/35
1079/1079 [==============================] - 9s 9ms/sample - loss: 1442087.6704
Epoch 16/35
1079/1079 [==============================] - 9s 8ms/sample - loss: 1043323.6124
Epoch 17/35
1079/1079 [==============================] - 8s 8ms/sample - loss: 712946.9689
Epoch 18/35
1079/1079 [==============================] - 8s 8ms/sample - loss: 408659.9425
Epoch 19/35
1079/1079 [==============================] - 9s 8ms/sample - loss: 407012.4942
Epoch 20/35
1079/1079 [==============================] - 9s 8ms/sample - loss: 396053.8325
Epoch 21/35
1079/1079 [==============================] - 8s 8ms/sample - loss: 396733.0584
Epoch 22/35
1079/1079 [==============================] - 8s 8ms/sample - loss: 420808.4667
Epoch 23/35
1079/1079 [==============================] - 8s 8ms/sample - loss: 400491.6283
Epoch 24/35
1079/1079 [==============================] - 9s 8ms/sample - loss: 405867.2980
Epoch 25/35
1079/1079 [==============================] - 9s 8ms/sample - loss: 414049.7230
Epoch 26/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 380021.3067
Epoch 27/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 408374.1941
Epoch 28/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 378242.5232
Epoch 29/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 378265.6125
Epoch 30/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 427112.9139
Epoch 31/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 440628.6283
Epoch 32/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 371036.2166
Epoch 33/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 388836.2415
Epoch 34/35
1079/1079 [==============================] - 8s 7ms/sample - loss: 394667.3049
Epoch 35/35
```

```
1079/1079 [==============================] - 8s 8ms/sample - loss: 383025.5851
lstm: [642.431] 619.8, 497.2, 459.6, 658.6, 613.2, 637.6, 911.1
```




```
Saving figure Neurons400Decoder3
```



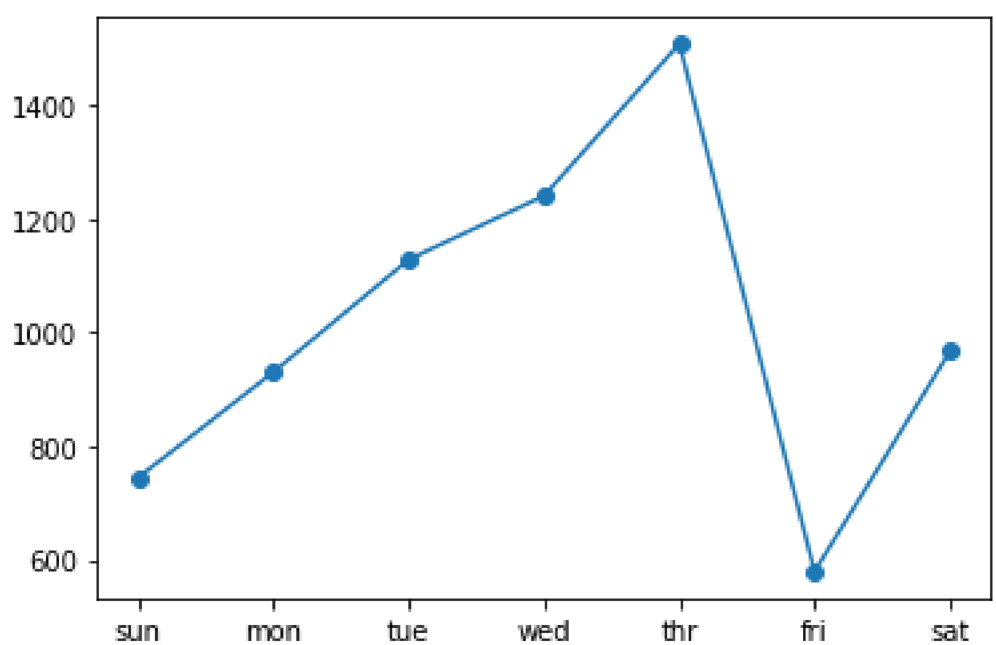
5.4.3 200 neurons, 35 epochs, 2e/3d, dropout=0, regularization=none

[24]:

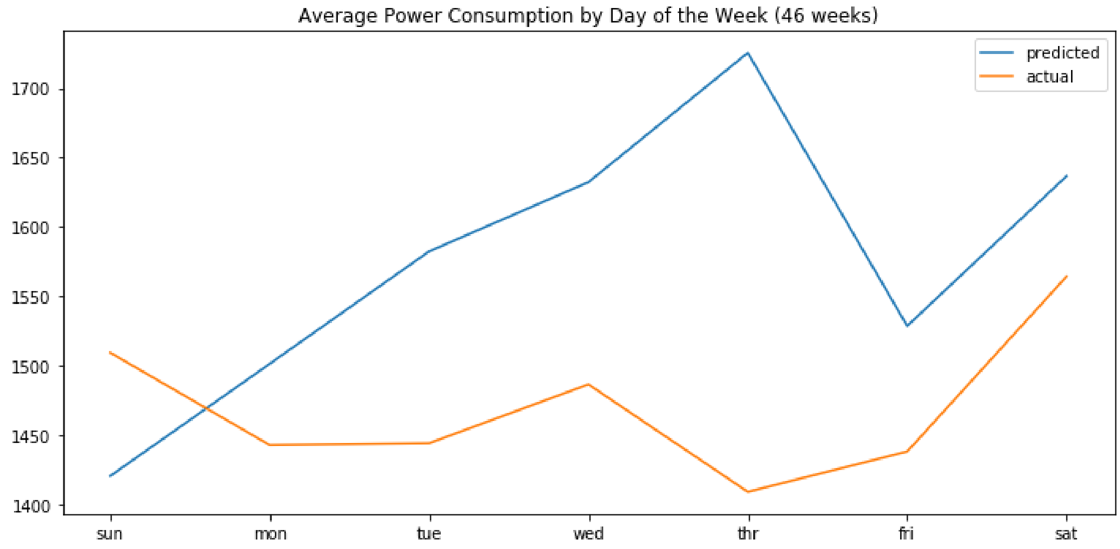
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 35, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
 ↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
 ↪n_features)))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(50, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
 ↪verbose=verbose)
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("Neurons200Decoder3")
pyplot.show()
```

```
Train on 1079 samples
Epoch 1/35
1079/1079 [==============================] - 3s 3ms/sample - loss:
707894145.3939
Epoch 2/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 94293044.7192
Epoch 3/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 26987750.6988
Epoch 4/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 24410645.7813
Epoch 5/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 10024179.9796
Epoch 6/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 7423889.5473
Epoch 7/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3553172.1066
Epoch 8/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2838227.3230
Epoch 9/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 5861982.9568
Epoch 10/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 685636.8706
Epoch 11/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 409219.8582
Epoch 12/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 417782.0028
Epoch 13/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 486914.6483
Epoch 14/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2084411.6520
Epoch 15/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 476126.8503
Epoch 16/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 705277.6773
Epoch 17/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 623076.9990
Epoch 18/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 412000.7300
Epoch 19/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 344465.7478
Epoch 20/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 348548.9464
Epoch 21/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 401714.8907
Epoch 22/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 515632.4787
Epoch 23/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 563763.6064
```

```
Epoch 24/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 375826.2583
Epoch 25/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 346237.0240
Epoch 26/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 12443716.0957
Epoch 27/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 624149.2333
Epoch 28/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 748606.8209
Epoch 29/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 538422.1625
Epoch 30/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 450235.5433
Epoch 31/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 378405.7483
Epoch 32/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 391837.2130
Epoch 33/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 402442.3420
Epoch 34/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 436096.0206
Epoch 35/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 449057.7634
lstm: [1054.489] 745.4, 930.9, 1128.9, 1241.0, 1508.3, 579.9, 967.3
```



```
Saving figure Neurons200Decoder3
```



5.4.4 200 neurons, 35 epochs, 2e/4d, dropout=0, regularization=none

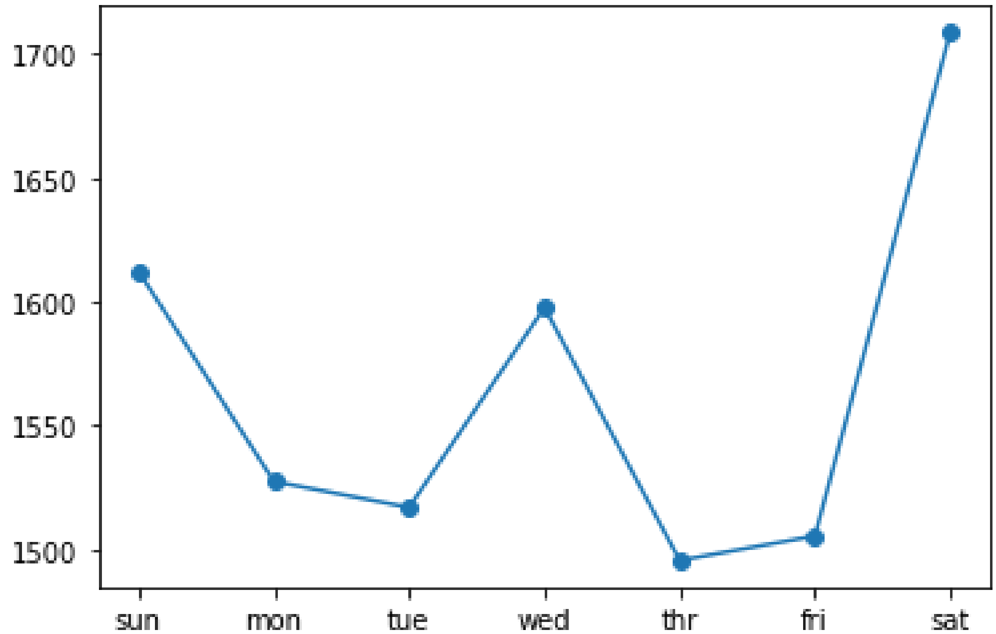
[25]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 35, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
 ↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
 ↪n_features)))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(50, activation='relu')))
        model.add(TimeDistributed(Dense(10, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
 ↪verbose=verbose)
```

```python
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("Neurons200Decoder4")
pyplot.show()
```

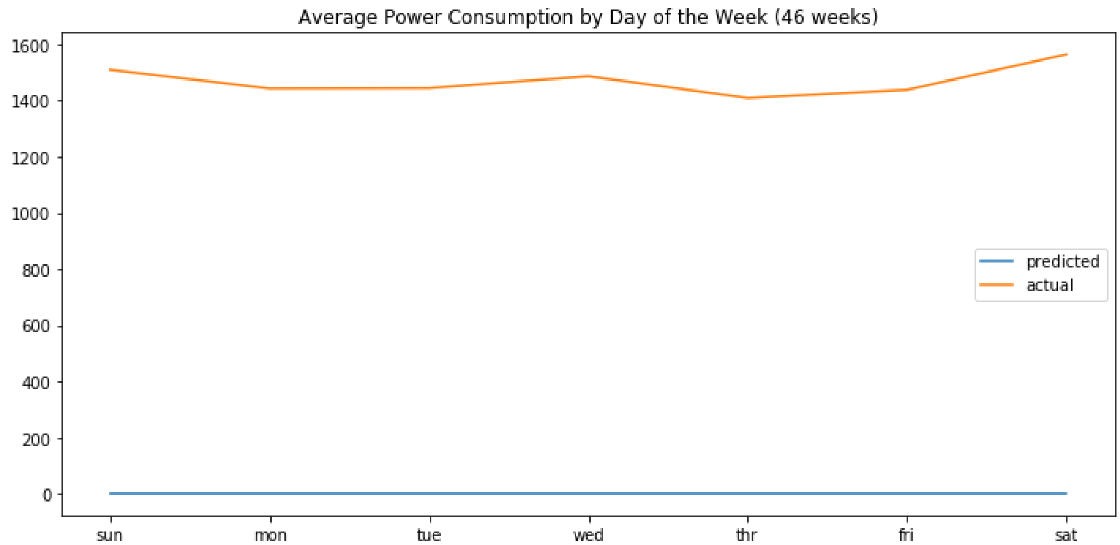
```
Train on 1079 samples
Epoch 1/35
1079/1079 [==============================] - 4s 3ms/sample - loss: 86268628.0946
Epoch 2/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1458124.6680
Epoch 3/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 757906.1076
Epoch 4/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 597573.4613
Epoch 5/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 412961.4434
Epoch 6/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 434960.3990
Epoch 7/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 761531.4112
Epoch 8/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1168756.2475
0s - loss: 109
Epoch 9/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 742192.9487
Epoch 10/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 726680.1602
Epoch 11/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1817271.8937
```

```
Epoch 12/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2908921.9826
Epoch 13/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2773927.2023
Epoch 14/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2773658.5945
Epoch 15/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2773397.3281
Epoch 16/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2773143.1295
Epoch 17/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2772894.5308
Epoch 18/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2772651.4071
Epoch 19/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2772411.7539
Epoch 20/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2772175.1138
Epoch 21/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2771941.1937
Epoch 22/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2771709.9036
0s - loss:
Epoch 23/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2771480.5313
Epoch 24/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2771253.1545
Epoch 25/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2771027.4509
Epoch 26/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2770803.1082
Epoch 27/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2770579.8610
Epoch 28/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2770357.8356
Epoch 29/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2770136.9824
Epoch 30/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2769917.0968
Epoch 31/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2769697.5053
Epoch 32/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2769479.0630
Epoch 33/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2769261.3406
Epoch 34/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2769044.4516
Epoch 35/35
```

```
1079/1079 [==============================] - 2s 2ms/sample - loss: 2768828.0462
lstm: [1567.987] 1612.4, 1527.5, 1517.0, 1597.7, 1495.5, 1505.4, 1708.9
```




```
Saving figure Neurons200Decoder4
```



5.5 Dropout Experiment

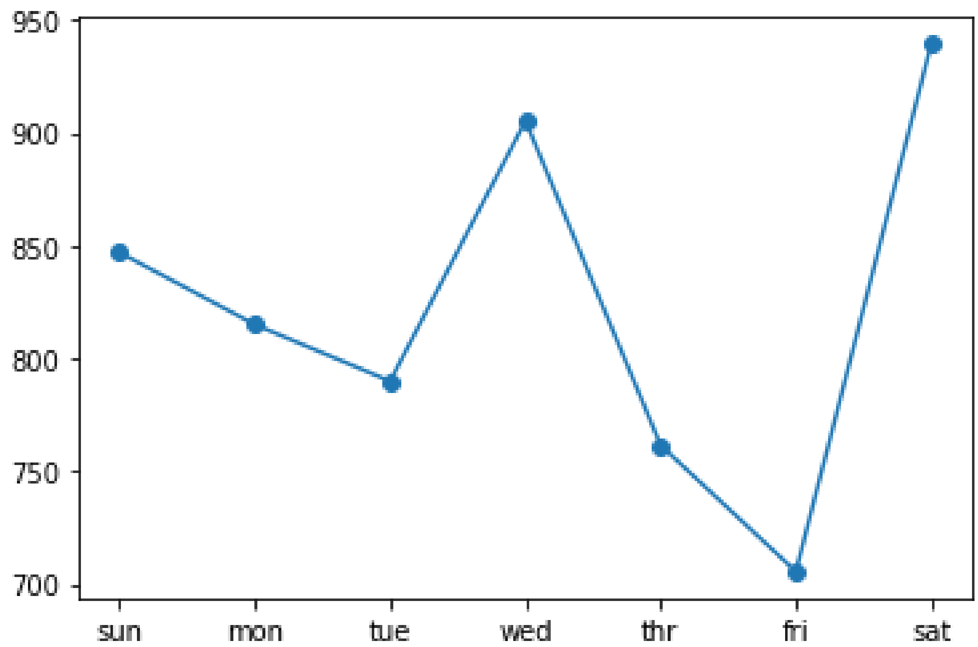
5.5.1 200 neurons, 35 epochs, 2e/3d, dropout=.2, regularization=none

[26]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 35, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
 ↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
 ↪n_features), dropout=0.2))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(50, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
 ↪verbose=verbose)
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("Dropout2")
```

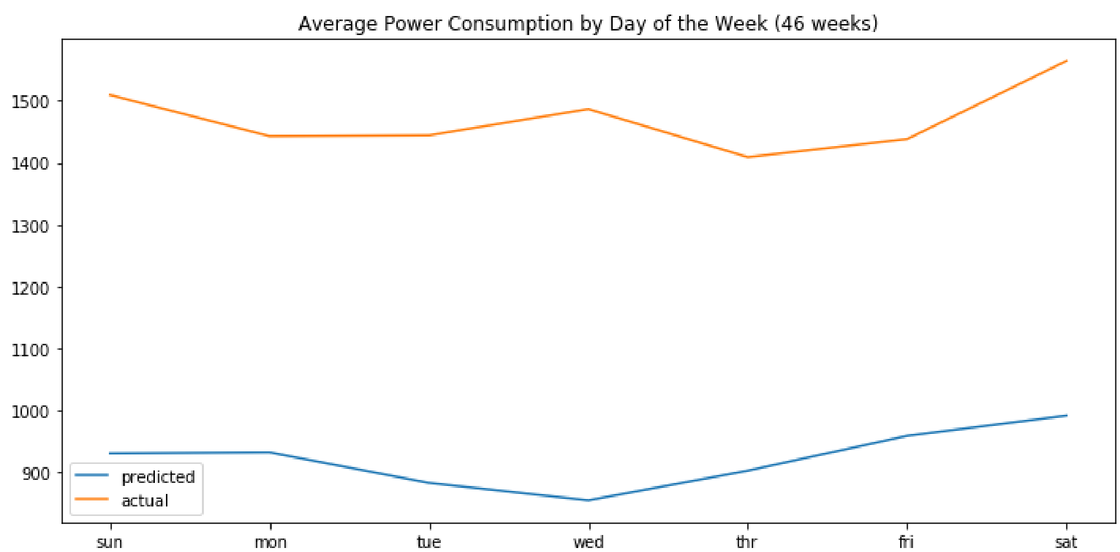
```
pyplot.show()
```

```
Train on 1079 samples
Epoch 1/35
1079/1079 [==============================] - 3s 3ms/sample - loss:
336494132.9082
Epoch 2/35
1079/1079 [==============================] - 2s 2ms/sample - loss:
146973501.5829
Epoch 3/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 72940269.0371
Epoch 4/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 41429232.5283
Epoch 5/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 17697709.1047
Epoch 6/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 13939509.3068
Epoch 7/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 19310171.3559
Epoch 8/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 5419528.3957
Epoch 9/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3721178.4893
Epoch 10/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 7574442.5058
Epoch 11/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 6635565.2845
Epoch 12/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 40219594.2261
Epoch 13/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 12040307.5412
Epoch 14/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4101439.4092
Epoch 15/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3360666.3647
Epoch 16/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3201969.8781
1s - loss: 406 - ETA: 1s -
Epoch 17/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3042377.2397
Epoch 18/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 8270607.1353
Epoch 19/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4542311.1657
Epoch 20/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 9778778.3424
Epoch 21/35
```

```
1079/1079 [==============================] - 2s 2ms/sample - loss: 4722361.8577
0s - loss: 50024
Epoch 22/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1790007.3007
Epoch 23/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1380768.2443
Epoch 24/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1135490.7287
Epoch 25/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 46158421.5630
Epoch 26/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 31392845.1622
Epoch 27/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 10857038.1872
Epoch 28/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2704954.6543
Epoch 29/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2646432.5132
Epoch 30/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1683931.5591
Epoch 31/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 989042.3570:
0s - loss: 997631.4
Epoch 32/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 816400.0897
Epoch 33/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 537434.5203
Epoch 34/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 544228.1121
Epoch 35/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 634798.5151
lstm: [826.815] 847.0, 815.0, 789.9, 905.3, 761.5, 705.5, 939.5
```



Saving figure Dropout2



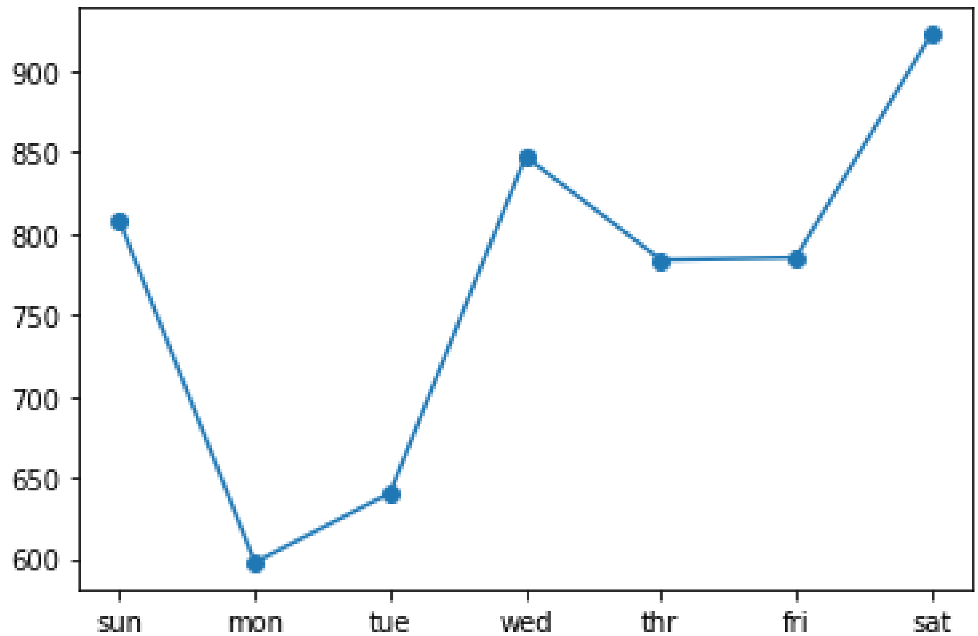
5.5.2 200 neurons, 50 epochs, 2e/3d, dropout=.2, regularization=none

[27]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 50, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
 ↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
 ↪n_features), dropout=0.2))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(50, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
 ↪verbose=verbose)
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("Dropout2Epochs50")
pyplot.show()
```

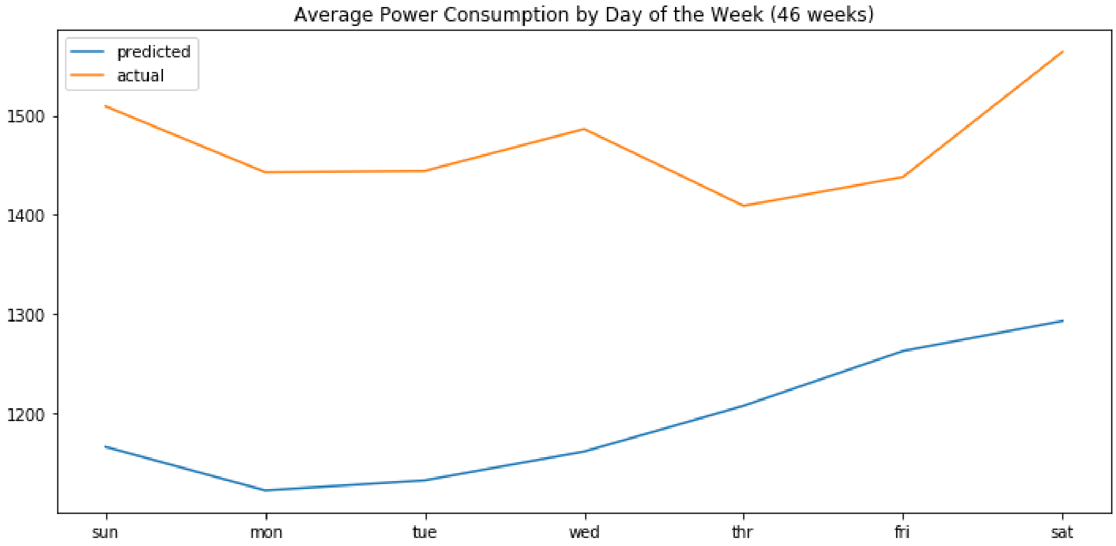
```
Train on 1079 samples
Epoch 1/50
1079/1079 [==============================] - 3s 3ms/sample - loss:
1995901374.8434
Epoch 2/50
1079/1079 [==============================] - 2s 2ms/sample - loss:
267867988.7303
Epoch 3/50
1079/1079 [==============================] - 2s 2ms/sample - loss:
739245055.0806
Epoch 4/50
1079/1079 [==============================] - 2s 2ms/sample - loss:
245106326.9509
Epoch 5/50
1079/1079 [==============================] - 2s 2ms/sample - loss:
40631119.46250s - loss: 426
Epoch 6/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 30002704.7868
Epoch 7/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 65485354.1307
Epoch 8/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 10732740.1274
Epoch 9/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 5415792.1353
Epoch 10/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 14014867.1886
Epoch 11/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 7663816.9911
Epoch 12/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 718103.0678
Epoch 13/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 633914.0090
Epoch 14/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 786763.46211s
- loss: 764
Epoch 15/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1411347.8962
Epoch 16/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 852403.8428
Epoch 17/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 595046.5699
Epoch 18/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 530112.9220
Epoch 19/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 498598.7812
Epoch 20/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 621022.8597
Epoch 21/50
```

```
1079/1079 [==============================] - 2s 2ms/sample - loss: 651593.8879
Epoch 22/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 698155.9774
Epoch 23/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 2221597.4993
Epoch 24/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 2093521.1084
Epoch 25/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1465749.5477
Epoch 26/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 2187845.2632
Epoch 27/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1838889.1931
Epoch 28/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 3462679.2121
Epoch 29/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 3609606.9662
Epoch 30/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 2864259.1513
Epoch 31/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1617354.8583
Epoch 32/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1122545.9412
Epoch 33/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 932504.4332
Epoch 34/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 912273.7871
Epoch 35/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1529346.4818
Epoch 36/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1131417.6081
Epoch 37/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 2795513.0807
Epoch 38/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1214362.6463
Epoch 39/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1397560.5142
Epoch 40/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1075900.3716
Epoch 41/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1469175.4835
Epoch 42/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1996135.2405
Epoch 43/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 2492841.9985
Epoch 44/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 5372248.1664
Epoch 45/50
```

```
1079/1079 [==============================] - 2s 2ms/sample - loss: 1432699.0987
Epoch 46/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1177905.0623
Epoch 47/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 1082110.5191
Epoch 48/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 3216760.5673
Epoch 49/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 919339.2563
Epoch 50/50
1079/1079 [==============================] - 2s 2ms/sample - loss: 810732.4088
lstm: [776.465] 807.7, 597.8, 640.5, 847.3, 783.9, 785.5, 922.4
```




```
Saving figure Dropout2Epochs50
```



5.5.3 200 neurons, 35 epochs, 2e/3d, dropout=.5, regularization=none

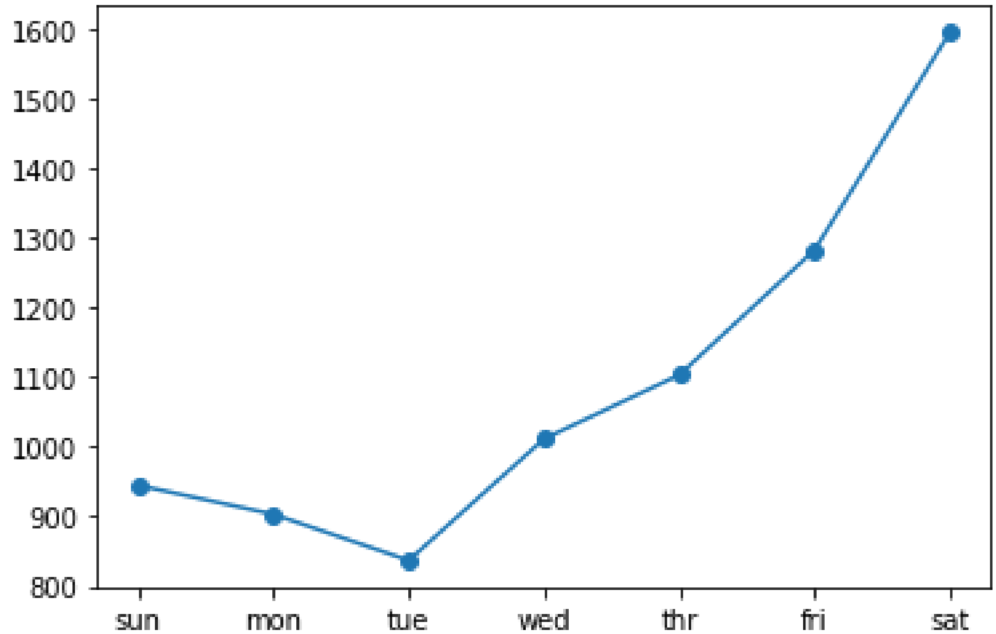
[28]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 35, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
↪n_features), dropout=0.5))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(50, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
↪verbose=verbose)
        return model

# evaluate model and get scores
```

```
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("Dropout5")
pyplot.show()
```

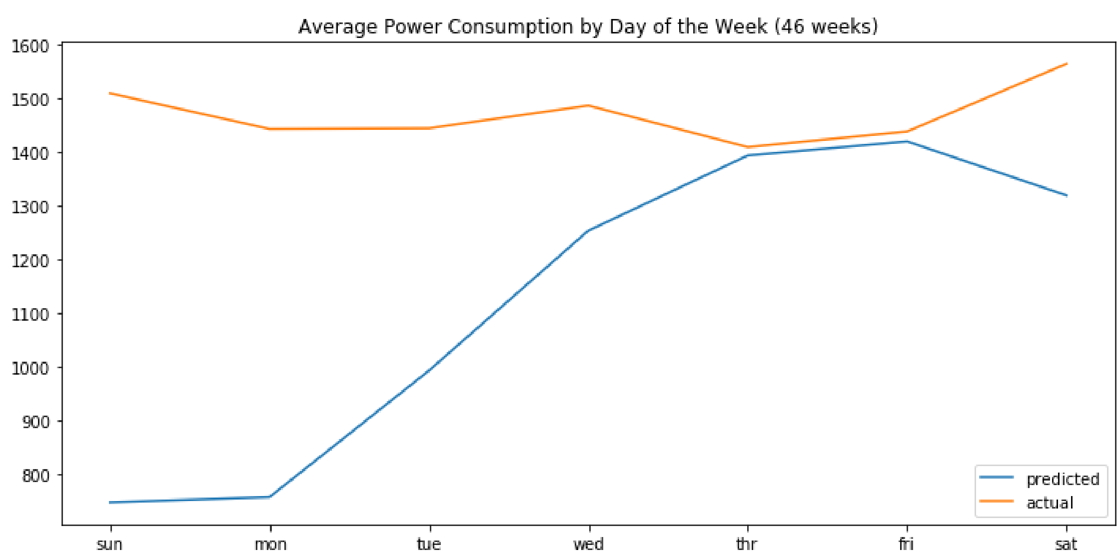
```
Train on 1079 samples
Epoch 1/35
1079/1079 [==============================] - 4s 3ms/sample - loss:
786056076.7618
Epoch 2/35
1079/1079 [==============================] - 2s 2ms/sample - loss:
445256146.4467
Epoch 3/35
1079/1079 [==============================] - 2s 2ms/sample - loss:
104598025.8703
Epoch 4/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 90512104.1372
Epoch 5/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 41953099.5292
Epoch 6/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 65098452.8193
Epoch 7/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 37980035.1066
Epoch 8/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 72341442.6006
Epoch 9/35
1079/1079 [==============================] - 2s 2ms/sample - loss:
137443572.9787
Epoch 10/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 48438639.7201
Epoch 11/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 7620573.9618
```

```
Epoch 12/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3142702.1475
Epoch 13/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4733105.2690
Epoch 14/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4667252.1209
Epoch 15/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4625052.4690
Epoch 16/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4935781.7224
Epoch 17/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2862752.0770
Epoch 18/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 9817330.2451
Epoch 19/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 9431127.3960
Epoch 20/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 11743353.2634
Epoch 21/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 13702035.9949
Epoch 22/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3370478.3777
Epoch 23/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2473446.7785
Epoch 24/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4256765.7303
Epoch 25/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2020270.9269
Epoch 26/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2992985.6419
Epoch 27/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2686251.5642
Epoch 28/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2307016.8797
Epoch 29/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2188158.6637
Epoch 30/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1716343.0346
Epoch 31/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3568900.0199
Epoch 32/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2018180.2609
1s -
Epoch 33/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1343906.4578
Epoch 34/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2519370.7944
Epoch 35/35
```

```
1079/1079 [==============================] - 2s 2ms/sample - loss: 1611114.2024
lstm: [1123.419] 944.6, 903.0, 836.8, 1011.5, 1103.0, 1282.3, 1594.5
```




```
Saving figure Dropout5
```



5.6 Regularization Experiment

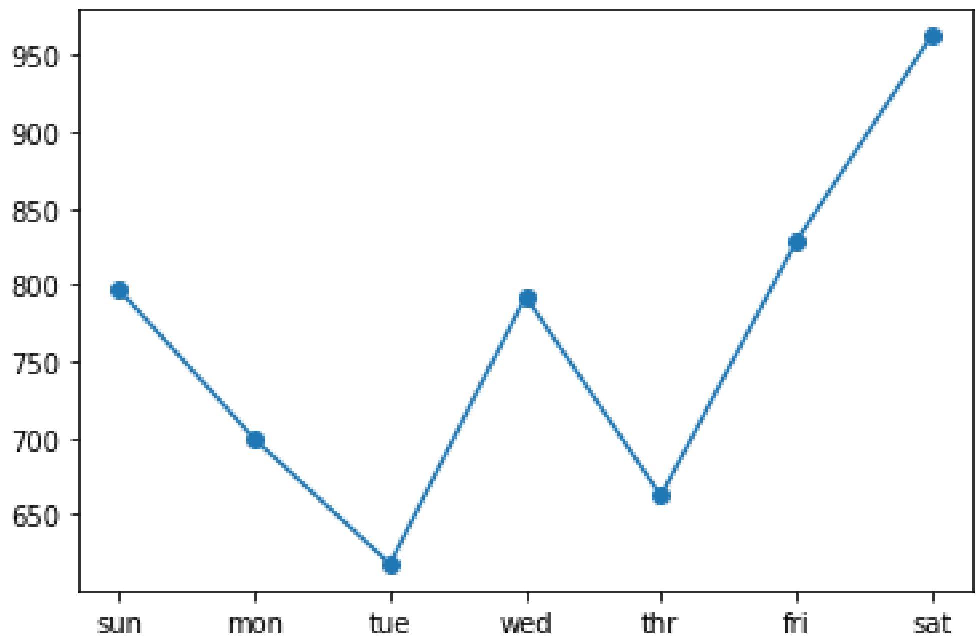
5.6.1 200 neurons, 35 epochs, 2e/3d, dropout=0, kernel regularization=l1(0.01)

[29]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 35, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
↪n_features), kernel_regularizer='l1'))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(50, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
↪verbose=verbose)
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("RegularizationL1")
```

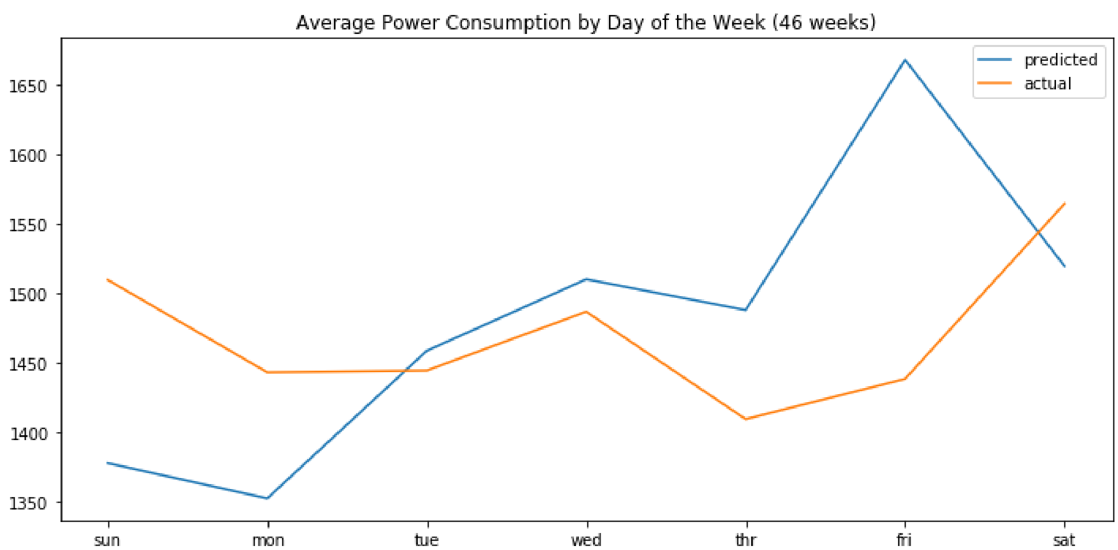
```
pyplot.show()
```

```
Train on 1079 samples
Epoch 1/35
1079/1079 [==============================] - 3s 3ms/sample - loss:
499188318.2280 0s - loss: 593886859.555 - ETA: 0s - loss: 57581303
Epoch 2/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 86749841.0528
Epoch 3/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 69373701.0408
Epoch 4/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 19016579.9147
Epoch 5/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 13456029.5894
Epoch 6/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 17706760.7488
Epoch 7/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 23702944.7859
Epoch 8/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 8017827.2034
Epoch 9/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3669562.3377
Epoch 10/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3840063.0491
Epoch 11/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3439732.7878
1s
Epoch 12/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 2791639.8415
Epoch 13/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 5442931.2244
Epoch 14/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4604769.0424
Epoch 15/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 719007.3466
Epoch 16/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 340195.4933
Epoch 17/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 324290.4778
Epoch 18/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 398026.2073
Epoch 19/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 320154.4936
Epoch 20/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 323730.6725
Epoch 21/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 803102.8077
```

```
Epoch 22/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 8004000.3368
Epoch 23/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4128955.9886
Epoch 24/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 4458794.1562
Epoch 25/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 5039320.4989
Epoch 26/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 432919.6908
Epoch 27/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 646364.1550
Epoch 28/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1697684.7815
Epoch 29/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1435750.7033
Epoch 30/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1511628.5215
Epoch 31/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1006345.2500
Epoch 32/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 753314.7709
Epoch 33/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 590069.1573
Epoch 34/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 596061.1783
Epoch 35/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 777020.5209
lstm: [773.021] 796.4, 699.1, 617.5, 792.2, 662.1, 828.2, 962.5
```




```
Saving figure RegularizationL1
```

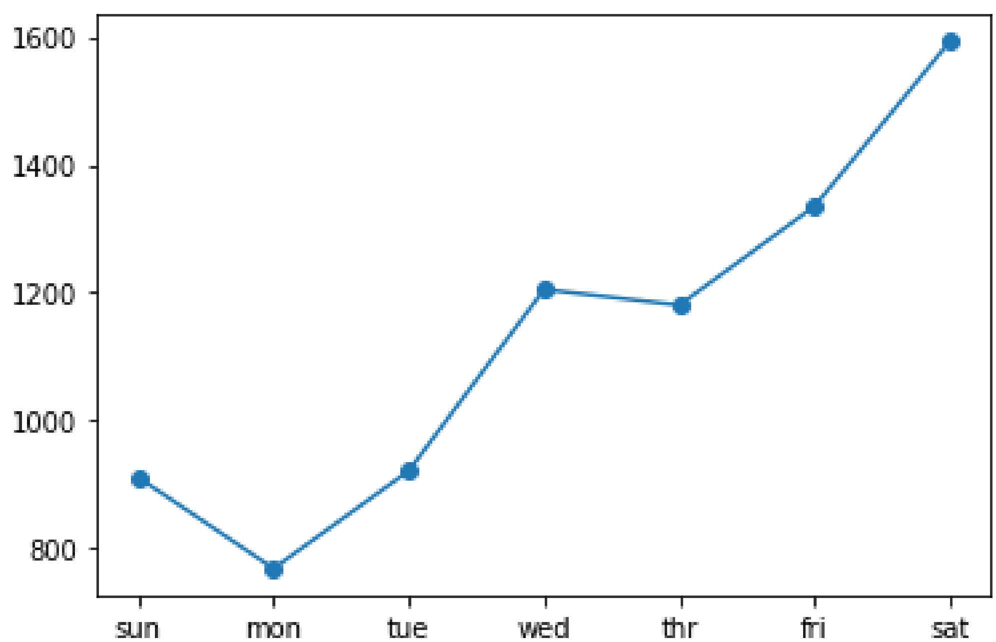


5.6.2 200 neurons, 35 epochs, 2e/3d, dropout=0, kernel regularization=l2(0.01)

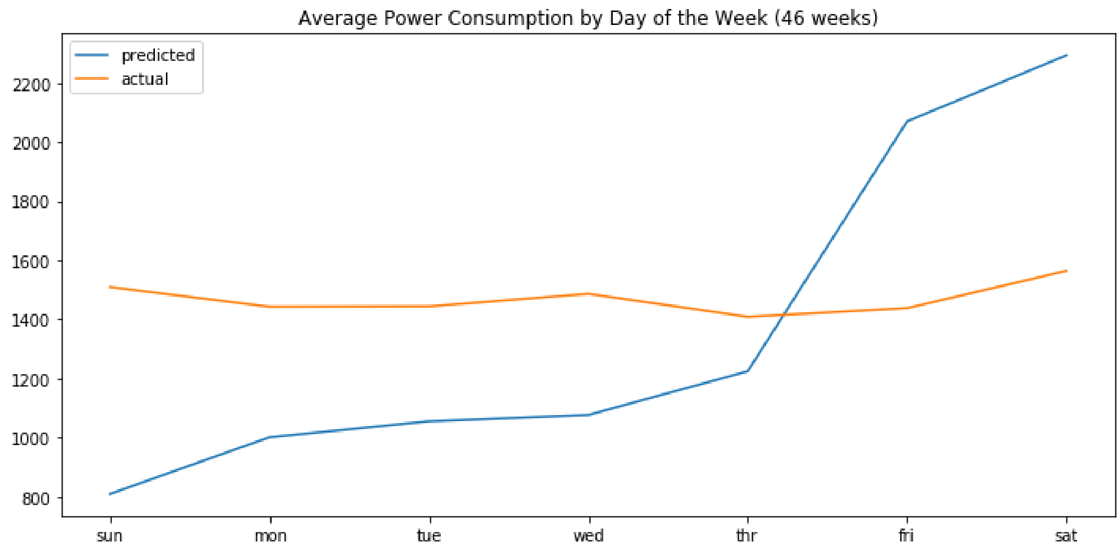
[30]:
```
# train the model
def build_model(train, n_input):
        # prepare data
        train_x, train_y = to_supervised(train, n_input)
        # define parameters
        verbose, epochs, batch_size = 1, 35, 16
        n_timesteps, n_features, n_outputs = train_x.shape[1], train_x.
 ↪shape[2], train_y.shape[1]
        # reshape output into [samples, timesteps, features]
        train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
        # define model
        model = Sequential()
        model.add(LSTM(200, activation='relu', input_shape=(n_timesteps,␣
 ↪n_features), kernel_regularizer='l2'))
        model.add(RepeatVector(n_outputs))
        model.add(LSTM(200, activation='relu', return_sequences=True))
        model.add(TimeDistributed(Dense(100, activation='relu')))
        model.add(TimeDistributed(Dense(50, activation='relu')))
        model.add(TimeDistributed(Dense(1)))
        model.compile(loss='mse', optimizer='adam')
        # fit network
        model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size,␣
 ↪verbose=verbose)
        return model

# evaluate model and get scores
score, scores, predictions = evaluate_model(train, test, n_input)
# summarize scores
summarize_scores('lstm', score, scores)
predictions_df = pd.DataFrame.from_records(predictions)
predicted_values = predictions_df.mean(axis=0)

# plot scores
days = ['sun', 'mon', 'tue', 'wed', 'thr', 'fri', 'sat']
pyplot.plot(days, scores, marker='o', label='lstm')
pyplot.show()

# plot actual vs predicted
pyplot.figure(figsize=(10,5))
pyplot.plot(days, predicted_values, label='predicted')
pyplot.plot(days, test_values, label='actual')
pyplot.legend()
pyplot.title("Average Power Consumption by Day of the Week (46 weeks)")
save_fig("RegularizationL2")
pyplot.show()
```

```
Train on 1079 samples
Epoch 1/35
1079/1079 [==============================] - 4s 3ms/sample - loss:
390137304.3855
Epoch 2/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 83415887.4217
Epoch 3/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 51482507.8109
Epoch 4/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 48847138.7044
Epoch 5/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 7832066.4022
Epoch 6/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1686235.4233
Epoch 7/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3250003.2694
Epoch 8/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 412364.3508
Epoch 9/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 411216.2292
Epoch 10/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 632167.5381
Epoch 11/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 937123.5877
Epoch 12/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 573836.1207
Epoch 13/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 435778.1305
Epoch 14/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 456151.6732
Epoch 15/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 420654.9846
Epoch 16/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 484272.6482
Epoch 17/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 465339.2096
Epoch 18/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 413044.2812
Epoch 19/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 415085.1830
Epoch 20/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 526066.1262
Epoch 21/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 411028.7176
Epoch 22/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 451853.9906
Epoch 23/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 408929.1913
```

```
Epoch 24/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 464919.3236
Epoch 25/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 508247.9908
Epoch 26/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 537332.0120
Epoch 27/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 440161.6779
Epoch 28/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 461588.4599
Epoch 29/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 461890.7542
Epoch 30/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 442068.8170
Epoch 31/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 458724.3975
Epoch 32/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 6629323.8445
Epoch 33/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 1207304.5990
Epoch 34/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 11369418.3313
Epoch 35/35
1079/1079 [==============================] - 2s 2ms/sample - loss: 3200509.9328
0s - loss:
lstm: [1160.710] 910.4, 767.5, 920.7, 1204.8, 1180.2, 1335.0, 1593.3
```



Saving figure RegularizationL2




```
[ ]:
```