Model-Experiments

March 30, 2022

[1]:
```python
import pandas as pd
import numpy as np
from sklearn.metrics import *
from sklearn.model_selection import train_test_split

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
```

1 Loading Datasets

[2]:
```python
# Loading datas
labels = ['Lagged', 'MA', 'WMA', 'MA-Lagged', 'WMA-Lagged'] # names of each␣
 ↪datasets

def load_datasets():
    """
    Excel files for each dataset are read into a
    dataframe an stored in a dictionary for easy
    access and use
    """
    datasets = dict()
    for lb in labels:
        new_df = pd.read_excel(f"River-Data-{lb}.xlsx")
        new_df.drop(["Unnamed: 0"], axis=1, inplace=True)
        datasets[lb] = new_df

    return datasets

data = load_datasets() # a dataframe for each dataset is stored in a dictionary␣
 ↪called data
```

2 Utility Functions

Standardising and Unstandardising Values

[3]:

```python
# Utility Functions
## Functions for standardising and unstandardising values
def standardise_columns(df, cols):
    """
    This function works with dataframes to standardise values
    in multiple columns to the range [0.1, 0.9]
    """
    subset_df = df[cols]
    subset_df = 0.8 * ((subset_df - subset_df.min()) / (subset_df.max() -␣
 ↪subset_df.min())) + 0.1
    return subset_df

def unstandardise_columns(df, cols, max_val, min_val):
    """
    This function works with numpy arrays to destandardise values
    in multiple columns
    """
    subset_df = df[cols]
    subset_df = ((subset_df - subset_df.min()) / 0.8) * (max_val - min_val) +␣
 ↪min_val
    return subset_df

def standardise_value(x, max_val, min_val):
    """
    This function works with numpy arrays to standardise values
    in multiple arrays to the range [0.1, 0.9]
    """
    return 0.8 * ((x - min_val)) / (max_val - min_val) + 0.1

def unstandardise_value(x, max_val, min_val):
    """
    This function works with numpy arrays to destandardise values
    in multiple arrays
    """
    return ((x - 0.1) / 0.8) * (max_val - min_val) + min_val
```

Plotting

[4]:

```python
## Plotting functions
def plot_correlation_matrix(corr_data, title, figsize=(16,6), mask=False):
    """
    Utility function for plotting a correlation heatmap of a given feature set
    """
    if mask:
        mask = np.triu(np.ones_like(corr_data, dtype=bool))
    plt.figure(figsize=figsize, dpi=500)
    heatmap = sns.heatmap(corr_data, vmin=-1, vmax=1, annot=True, mask=mask)
    heatmap.set_title(title)
```

```
    plt.show()

def plot_predictions(preds_df, standardised=False):
    """
    Utiltity function for plotting model predictions against actual value
    """
    preds_col = "Predicted Values"
    vals_col = "Actual Values"
    if standardised:
        preds_col += " (Standardised)"
        vals_col += " (Standardised)"

    line_plt = px.line(preds_df, y=vals_col)
    scatter_plt = px.scatter(preds_df, y=preds_col,␣
 ↪color_discrete_sequence=["#ff0000"])

    go.Figure(line_plt.data + scatter_plt.data, layout={"title": "Actual vs␣
 ↪Predicted Values"}).show()
```

3 ANN Class

[5]:
```
# Basic ANN class for MLP models
class BasicAnn:
    def __init__(self, layers, max_st_val, min_st_val, activ_func="sigmoid"):
        self.layers = layers
        self.num_layers = len(layers)
        self.max_val = max_st_val
        self.min_val = min_st_val
        self.activ_func = activ_func

        weight_shapes = [(layers[i-1],layers[i]) for i in range(1, len(layers))]
        self.weights = {
            f"W{i+1}": np.random.standard_normal(s)/s[0]**0.5
            for i, s in enumerate(weight_shapes)
        } # weights are stored as matrices that are implemented as 2D numpy␣
 ↪arrays
        self.biases = {
            f"B{i+1}": np.random.randn(l,1)/l**0.5
            for i, l in enumerate(layers[1:])
        } # biases are also stored as matrices that are implemented as 2D numpy␣
 ↪arrays

    def activation(self, x):
        """
        Function to return value with the selected activation
        """
```

```
        if self.activ_func == "sigmoid":
            return 1/(1+np.exp(-x))
        elif self.activ_func == "tanh":
            return (np.exp(x)-np.exp(-x))/(np.exp(x)+np.exp(-x))
        elif self.activ_func == "relu":
            return x * (x > 0)
        elif self.activ_func == "linear":
            return x

    def activation_deriv(self, a):
        """
        Function to return value with the derivative of the selected activation
        """
        if self.activ_func == "sigmoid":
            return a * (1 - a)
        elif self.activ_func == "tanh":
            return 1 - a**2
        elif self.activ_func == "relu":
            return 1 * (a > 0)
        elif self.activ_func == "linear":
            return np.ones(a.shape)

    def train(self, features, targets, epochs=1000, learning_rate=0.1,␣
↪val_set=None):
        """
        Function will train the model using the standard backpropogation␣
↪algorithm
        and return a dataframe storing various error metrics for the model on␣
↪the
        training set and, possibly, a validation set if that is given
        """
        results = pd.DataFrame()
        real_targets = unstandardise_value(targets, self.max_val, self.min_val)
        num_targets = len(targets)

        for _ in range(epochs):
            # Forward pass
            activations = self.forward_pass(features)

            # Error calculation
            output_layer = activations[f"A{self.num_layers - 1}"]
            real_preds = unstandardise_value(output_layer, self.max_val, self.
↪min_val)
            error_data = { # storing error metrics for both standardised and␣
↪unstandardised data
                "mse": mean_squared_error(real_targets, real_preds),
```

```
                "rmse": mean_squared_error(real_targets, real_preds,␣
↪squared=False),
                "mae": mean_absolute_error(real_targets, real_preds),
                "r_sqr": r2_score(real_targets, real_preds),
                "st_mse": mean_squared_error(targets, output_layer),
                "st_rmse": mean_squared_error(targets, output_layer,␣
↪squared=False),
                "st_mae": mean_absolute_error(targets, output_layer),
                "st_r_sqr": r2_score(targets, output_layer)
            }

            if val_set:
                # if there is a validation set the prediction error of the model
                # on the validation set will be stored
                r, err = self.predict(val_set[0].to_numpy(), val_set[1].
↪to_numpy())
                error_data.update({f"val_{col}": err[col][0] for col in err.
↪columns})

            results = results.append(error_data, ignore_index=True)

            # Backward pass (backpropagation algorithm)
            deltas = self.compute_deltas(activations, targets, output_layer)
            self.update_weights(deltas, activations, features, num_targets,␣
↪learning_rate)

        return results

    def predict(self, test_inputs, st_actual_outputs, actual_outputs=None):
        """
        Runs a forward pass of the network with the newly configured weights
        and biases and returns a dataframe comparing the predicted values
        to actual values as well as a dataframe with various error metrics
        """
        # Forward pass
        activations = self.forward_pass(test_inputs)
        st_preds = activations[f"A{self.num_layers - 1}"]

        # Comparing predicted values with actual values
        if actual_outputs is None:
            actual_outputs = unstandardise_value(st_actual_outputs, self.
↪max_val, self.min_val)

        preds = unstandardise_value(st_preds, self.max_val, self.min_val)

        results = pd.DataFrame(
```

```
            data={
                "Actual Values": actual_outputs.flatten(),
                "Predicted Values": preds.flatten(),
                "Actual Values (Standardised)": st_actual_outputs.flatten(),
                "Predicted Values (Standardised)": st_preds.flatten(),
            }
        )

        # Error calculation
        results["Absolute Error"] = abs(results["Actual Values"] -␣
↪results["Predicted Values"])
        st_absolute_err = abs(results["Actual Values (Standardised)"] -␣
↪results["Predicted Values (Standardised)"])
        results["Absolute Error (Standardised Values)"] = st_absolute_err

        error_metrics = pd.DataFrame(data={
            "mse": [mean_squared_error(actual_outputs, preds)],
            "rmse": [mean_squared_error(actual_outputs, preds, squared=False)],
            "mae": [mean_absolute_error(actual_outputs, preds)],
            "r_sqr": [r2_score(actual_outputs, preds)],
            "st_mse": [mean_squared_error(st_actual_outputs, st_preds)],
            "st_rmse": [mean_squared_error(st_actual_outputs, st_preds,␣
↪squared=False)],
            "st_mae": [mean_absolute_error(st_actual_outputs, st_preds)],
            "st_r_sqr": [r2_score(st_actual_outputs, st_preds)]
        })

        return results, error_metrics

    def forward_pass(self, features):
        """
        Runs a forward pass of neural network through repeated
        multiplication of weights and bias matrices. Returns
        list of each activation layer including the output layer.
        """
        activation = self.activation(np.dot(features, self.weights["W1"]) +␣
↪self.biases["B1"].T)
        activations = {"A1": activation}
        for i in range(2, self.num_layers):
            activation = self.activation(np.dot(activation, self.
↪weights[f"W{i}"]) + self.biases[f"B{i}"].T)
            activations[f"A{i}"] = activation

        return activations

    def compute_deltas(self, activations, targets, output_layer):
        """
```

```
        Computes errors between layers for backprogation.
        Returns a dictionary of lists which contain the errors
        for each node in each layer.
        """
        output_err = targets - output_layer
        output_delta = output_err * self.activation_deriv(output_layer)
        deltas = {"dw1": output_delta}

        for i in range(self.num_layers - 1, 1, -1):
            dw = deltas[f"dw{self.num_layers - i}"]
            act = activations[f"A{i-1}"]
            w = self.weights[f"W{i}"]
            deltas[f"dw{self.num_layers - i + 1}"] = np.dot(dw, w.T) * self.
↪activation_deriv(act)

        return deltas

    def update_weights(self, deltas, activations, features, num_targets,␣
↪l_rate):
        """
        Updates weights and biases according to given errors, activations
        and the chosen learning rate
        """
        delta = deltas[f"dw{self.num_layers - 1}"]
        self.weights["W1"] += l_rate * (np.dot(features.T, delta)) / num_targets
        self.biases["B1"] += l_rate * (np.dot(delta.T, np.ones((num_targets,␣
↪1)))) / num_targets

        for i in range(2, self.num_layers):
            act = activations[f"A{i-1}"]
            dw = deltas[f"dw{self.num_layers - i}"]
            self.weights[f"W{i}"] += l_rate * (np.dot(act.T, dw)) / num_targets
            self.biases[f"B{i}"] += l_rate * np.dot(dw.T, np.ones((num_targets,␣
↪1))) / num_targets
```

Build, Train and Test ANN Model

[6]:
```
def build_train_test(feature_set, feature_cols, target_cols, layers=("auto",␣
↪1), activ_func="linear", epochs=1000, l_rate=0.1):
    """
    Function to build, train and test MLP models
    """
    # Splitting and standardising datasets to create standardised and␣
↪unstandardised
    # training, validation and testing sets.
    train_val_set, test_set = train_test_split(feature_set, test_size=0.2)
    st_train_val_set = standardise_columns(train_val_set, train_val_set.columns)
```

```python
    st_test_set = standardise_columns(test_set, test_set.columns)

    # Preparing features and targets for training and testing
    features = st_train_val_set[feature_cols]
    targets = st_train_val_set[target_cols]

    X_train, X_val, y_train, y_val = train_test_split(features, targets, test_size=0.25)
    X_test, y_test = st_test_set[feature_cols], st_test_set[target_cols]

    # Getting standardisation values for targets
    min_val = train_val_set[target_cols].min()[0]
    max_val = train_val_set[target_cols].max()[0]

    # Building model
    if layers[0] == "auto":
        # if the size of the input layer is not specified
        # then it will be set to the number of predictors
        layers = (len(feature_cols),) + layers[1:]

    ann = BasicAnn(layers, max_val, min_val, activ_func)

    # Training model
    training_results = ann.train(
        X_train.to_numpy(),
        y_train.to_numpy(),
        val_set=(X_val, y_val), # training with a validation set
        epochs=epochs,
        learning_rate=l_rate
    )

    # Predicting model
    prediction_results = ann.predict(
        X_test.to_numpy(),
        y_test.to_numpy(),
        actual_outputs=test_set[target_cols].to_numpy()
    )

    predictions, error_metrics = prediction_results[0], prediction_results[1]

    return {
        "training_results": training_results,
        "final_test_results": predictions,
        "error_metrics": error_metrics,
        "model": ann
    }
```

4 Selecting Features/Predictors

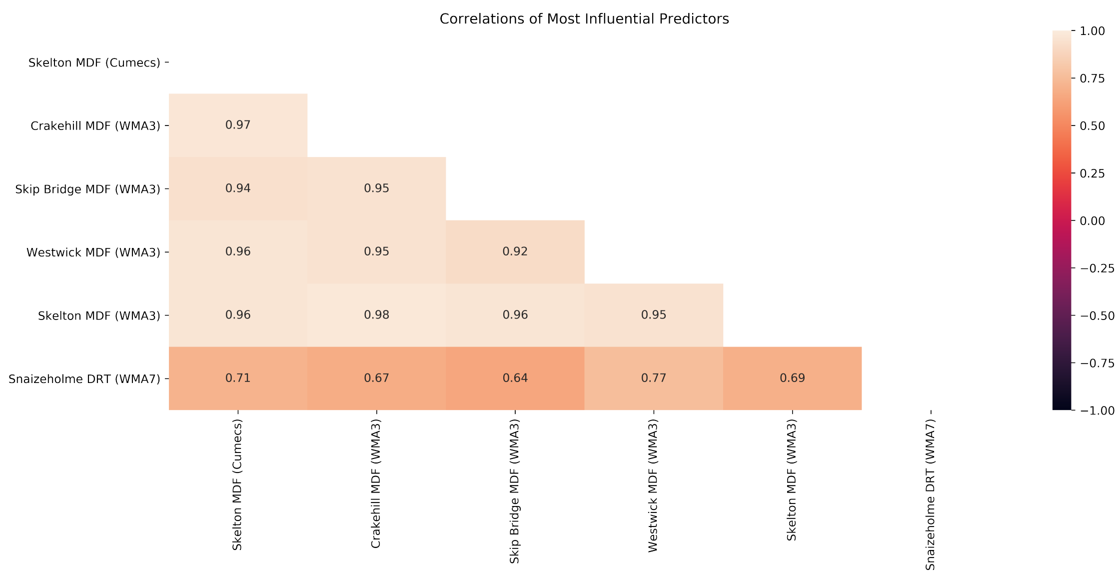
Building Feature Sets

[7]:
```python
# Function for building custom feature and target sets
def build_feature_set(*datasets):
    assert len(datasets) > 0, "No data sets entered"
    datasets = list(datasets)
    min_rows = min(d.shape[0] for d in datasets)

    for i, ds in enumerate(datasets):
        datasets[i] = ds.truncate(before=ds.shape[0]-min_rows).reset_index()
        datasets[i].drop(["index"], axis=1, inplace=True)

    merged_df = datasets[0].iloc[:, :2]
    for ds in datasets:
        merged_df = pd.concat([merged_df, ds.iloc[:, 2:]], axis=1)

    merged_cols = list(merged_df.columns)
    selected_cols = []

    for i in range(0, len(merged_cols), 2):
        format_str = f"{i+1}) {merged_cols[i]}"
        if i != len(merged_cols) - 1:
            second_part = f"{i+2}) {merged_cols[i+1]}"
            num_spaces = 50 - len(format_str)
            format_str += num_spaces*" " + second_part
        print(format_str)

    selected_indices = input("\nSelect columns: ")
    for index in selected_indices.split(","):
        if "-" in index:
            first_i, second_i = index.split("-")
            selected_cols += merged_cols[int(first_i) - 1: int(second_i)]
        else:
            selected_cols.append(merged_cols[int(index) - 1])

    return merged_df[selected_cols]
```

[8]:
```python
# 2,3-6,42
fs = build_feature_set(data['WMA'])
plot_correlation_matrix(fs.corr(), "Correlations of Most Influential␣
↪Predictors", mask=True)
```

```
1) Date                                           2) Skelton MDF (Cumecs)
3) Crakehill MDF (WMA3)                           4) Skip Bridge MDF (WMA3)
5) Westwick MDF (WMA3)                            6) Skelton MDF (WMA3)
7) Crakehill MDF (WMA4)                           8) Skip Bridge MDF (WMA4)
9) Westwick MDF (WMA4)                            10) Skelton MDF (WMA4)
```

```
11) Crakehill MDF (WMA5)
12) Skip Bridge MDF (WMA5)
13) Westwick MDF (WMA5)
14) Skelton MDF (WMA5)
15) Crakehill MDF (WMA6)
16) Skip Bridge MDF (WMA6)
17) Westwick MDF (WMA6)
18) Skelton MDF (WMA6)
19) Crakehill MDF (WMA7)
20) Skip Bridge MDF (WMA7)
21) Westwick MDF (WMA7)
22) Skelton MDF (WMA7)
23) Arkengarthdale DRT (WMA3)
24) East Cowton DRT (WMA3)
25) Malham Tarn DRT (WMA3)
26) Snaizeholme DRT (WMA3)
27) Arkengarthdale DRT (WMA4)
28) East Cowton DRT (WMA4)
29) Malham Tarn DRT (WMA4)
30) Snaizeholme DRT (WMA4)
31) Arkengarthdale DRT (WMA5)
32) East Cowton DRT (WMA5)
33) Malham Tarn DRT (WMA5)
34) Snaizeholme DRT (WMA5)
35) Arkengarthdale DRT (WMA6)
36) East Cowton DRT (WMA6)
37) Malham Tarn DRT (WMA6)
38) Snaizeholme DRT (WMA6)
39) Arkengarthdale DRT (WMA7)
40) East Cowton DRT (WMA7)
41) Malham Tarn DRT (WMA7)
42) Snaizeholme DRT (WMA7)

Select columns: 2,3-6,42
```



5 Training and Network Selection

Epochs

[9]:
```
target_cols = [fs.columns[0]]
feature_cols = list(fs.columns[1:])

epoch_tests = dict()
for i in range(1, 11):
    epoch_tests[f"Test-{i}"] = build_train_test(
```

```
        fs,
        feature_cols,
        target_cols,
        layers=("auto", 1),
        activ_func="linear",
        epochs=i*500
    )
```

- rmse -> root mean square error
- mae -> mean absolute error
- mse -> mean squared error
- r_sqr -> R-Squared (Coefficient of Determination)
- val_* -> error metric on validation set
- st_* -> error metric on unstandardised values

[10]:
```
# Testing number of epochs for training; between 800 and 1200 seems to be ideal
for i in range(1, 11):
    print(f"Model trained with {i*500} epochs", end=f"\n{'-'*100}\n")

    print("Final Training results")
    print(epoch_tests[f"Test-{i}"]["training_results"].iloc[-1, :4],␣
 ↪end=f"\n{'-'*100}\n")

    print("Final Validation results")
    print(epoch_tests[f"Test-{i}"]["training_results"].iloc[-1, 8:12],␣
 ↪end=f"\n{'-'*100}\n")

    print("Test Set Results")
    print(epoch_tests[f"Test-{i}"]["error_metrics"].iloc[0],␣
 ↪end=f"\n{'='*100}\n\n\n\n")

    ax = epoch_tests[f"Test-{i}"]["training_results"].plot(
        y=["rmse", "val_rmse"], title=f"Model Trained with {i*500} Epochs",
    )
    ax.set_xlabel("Epochs")
    ax.set_ylabel("Root Mean Squared Error (RMSE)")
```

```
Model trained with 500 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        14.250674
mse       608.270719
r_sqr       0.796728
rmse       24.663145
```

```
Name: 499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae       14.898253
val_mse      760.187084
val_r_sqr      0.768658
val_rmse      27.571490
Name: 499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         997.685762
rmse         31.586164
mae          19.087420
r_sqr         0.691045
st_mse        0.002737
st_rmse       0.052316
st_mae        0.031252
st_r_sqr      0.829853
Name: 0, dtype: float64
================================================================================
====================



Model trained with 1000 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        7.462920
mse      198.739946
r_sqr      0.940075
rmse      14.097516
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.262152
val_mse      232.112890
val_r_sqr      0.911425
val_rmse      15.235252
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         932.481105
rmse         30.536554
```

```
mae          18.012795
r_sqr         0.681143
st_mse        0.001910
st_rmse       0.043706
st_mae        0.020994
st_r_sqr      0.914481
Name: 0, dtype: float64
================================================================================
====================



Model trained with 1500 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae       12.254076
mse      401.123070
r_sqr      0.869136
rmse      20.028057
Name: 1499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae       11.954071
val_mse      430.115301
val_r_sqr      0.877540
val_rmse      20.739221
Name: 1499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         749.105870
rmse         27.369799
mae          16.441220
r_sqr         0.730054
st_mse        0.001777
st_rmse       0.042156
st_mae        0.025847
st_r_sqr      0.871320
Name: 0, dtype: float64
================================================================================
====================



Model trained with 2000 epochs
--------------------------------------------------------------------------------
```

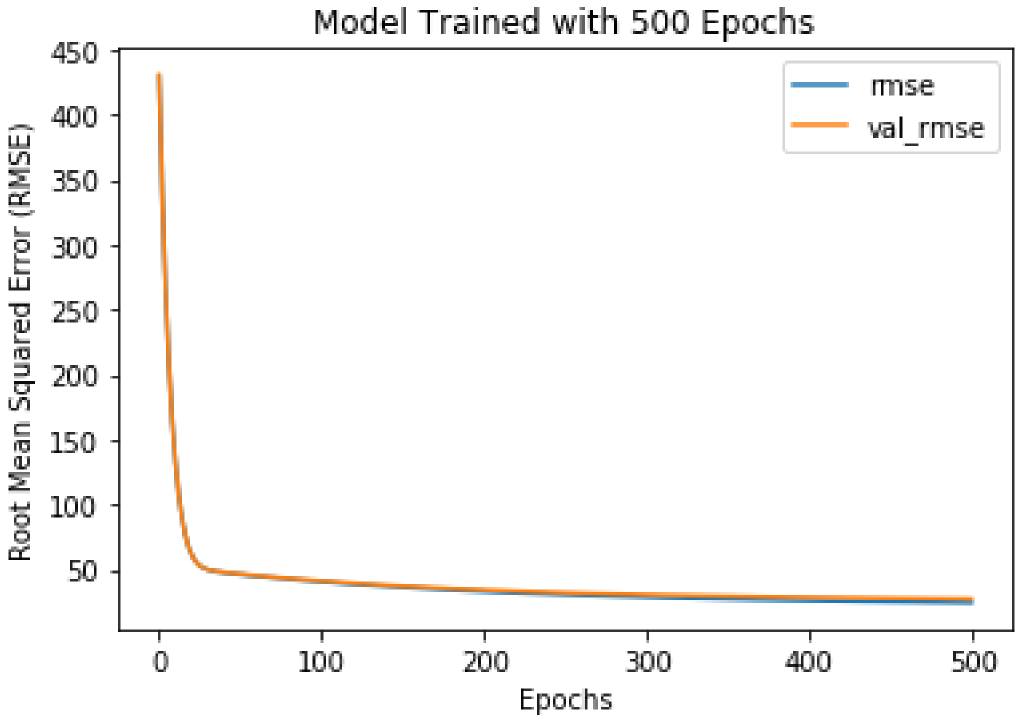
```
--------------------
Final Training results
mae        6.324813
mse      160.054810
r_sqr      0.944095
rmse      12.651277
Name: 1999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        5.521292
val_mse      113.716825
val_r_sqr      0.966952
val_rmse      10.663809
Name: 1999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1581.427295
rmse          39.767164
mae           24.011278
r_sqr          0.543158
st_mse         0.001004
st_rmse        0.031681
st_mae         0.019146
st_r_sqr       0.954978
Name: 0, dtype: float64
================================================================================
====================



Model trained with 2500 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        8.060659
mse      194.926005
r_sqr      0.941976
rmse      13.961590
Name: 2499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        7.718837
val_mse      153.990023
val_r_sqr      0.937434
val_rmse      12.409272
```

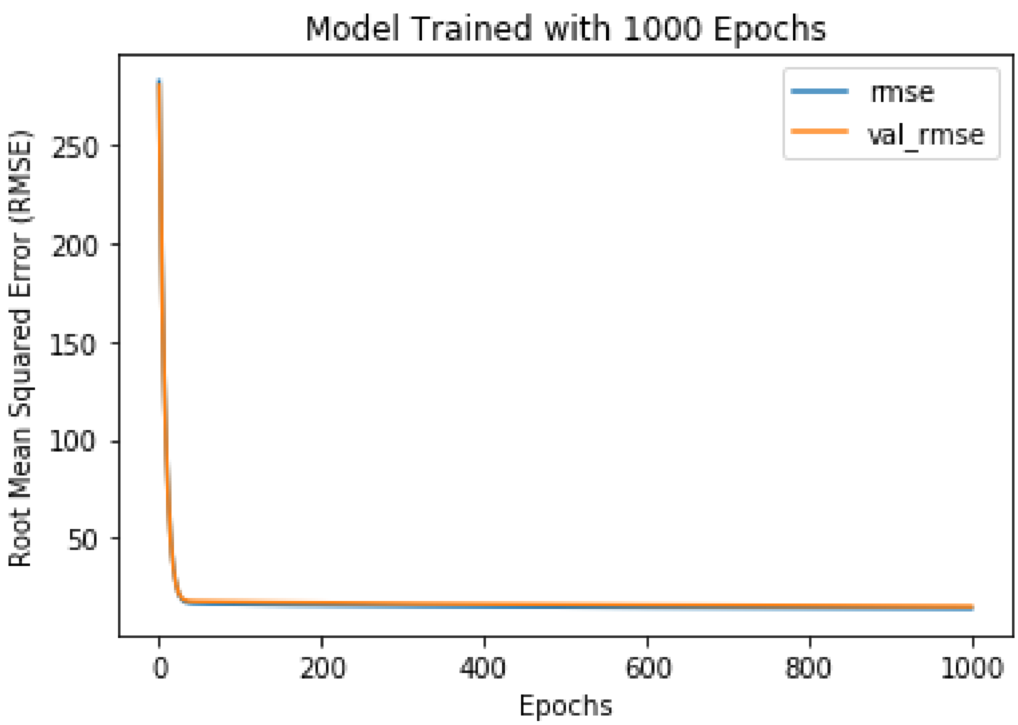
```
Name: 2499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1143.020080
rmse          33.808580
mae           20.808783
r_sqr          0.611482
st_mse         0.001003
st_rmse        0.031676
st_mae         0.017815
st_r_sqr       0.946819
Name: 0, dtype: float64
================================================================================
====================



Model trained with 3000 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        7.387638
mse      176.240466
r_sqr      0.949655
rmse      13.275559
Name: 2999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        7.344473
val_mse      149.975472
val_r_sqr      0.942712
val_rmse      12.246447
Name: 2999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1030.928267
rmse          32.108072
mae           19.467383
r_sqr          0.563387
st_mse         0.001600
st_rmse        0.040000
st_mae         0.021656
st_r_sqr       0.927151
Name: 0, dtype: float64
================================================================================
```

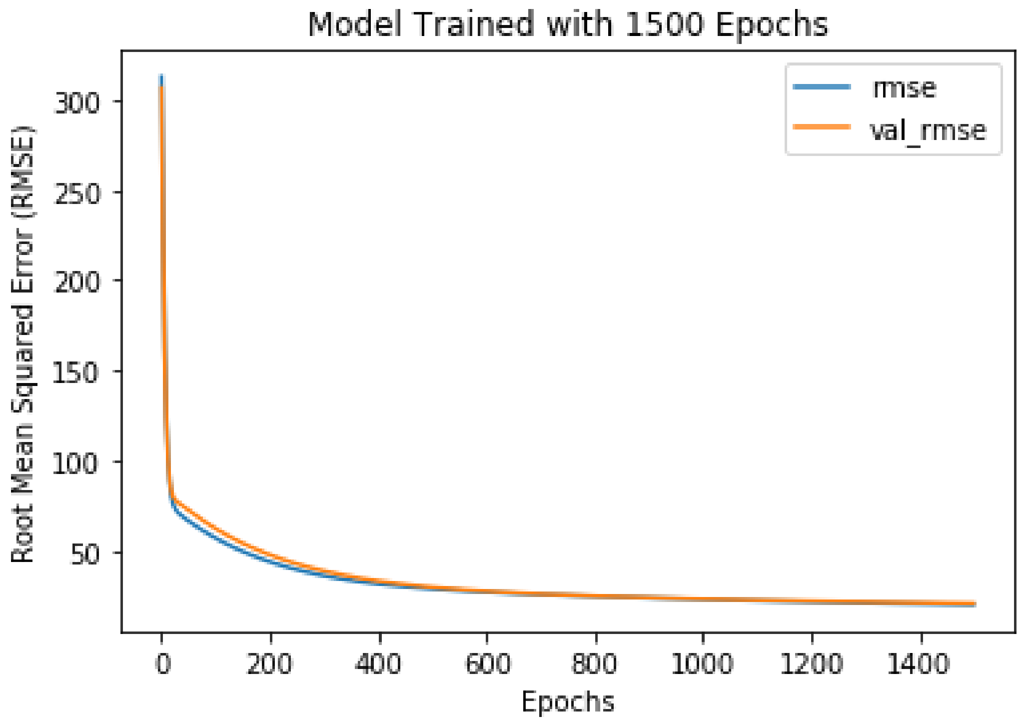
```
====================


Model trained with 3500 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        9.188253
mse      240.062707
r_sqr      0.916744
rmse      15.493957
Name: 3499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        9.977215
val_mse      284.290656
val_r_sqr      0.913355
val_rmse      16.860921
Name: 3499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         715.415637
rmse         26.747255
mae          15.717038
r_sqr         0.798475
st_mse        0.002235
st_rmse       0.047271
st_mae        0.026956
st_r_sqr      0.901951
Name: 0, dtype: float64
================================================================================
====================


Model trained with 4000 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        7.064710
mse      169.929624
r_sqr      0.944214
rmse      13.035706
Name: 3999, dtype: float64
--------------------------------------------------------------------------------
```

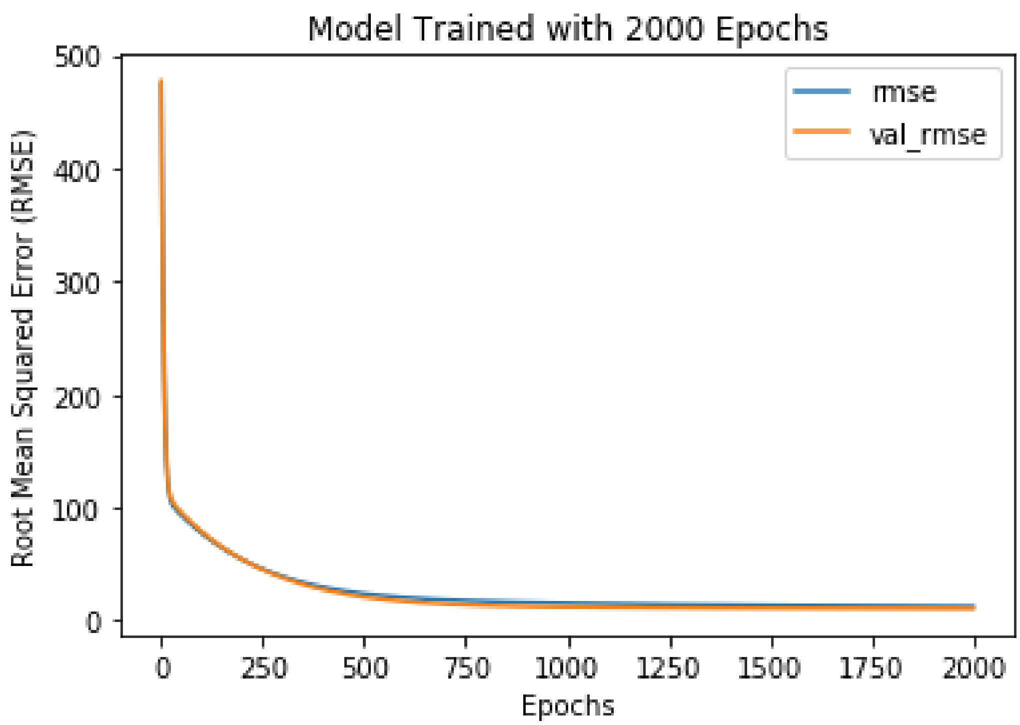
```
--------------------
Final Validation results
val_mae        6.893713
val_mse      181.037869
val_r_sqr      0.946676
val_rmse      13.455031
Name: 3999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         825.084728
rmse         28.724288
mae          15.858095
r_sqr         0.720780
st_mse        0.001050
st_rmse       0.032406
st_mae        0.017812
st_r_sqr      0.928742
Name: 0, dtype: float64
================================================================================
====================



Model trained with 4500 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        7.692935
mse      190.144104
r_sqr      0.932934
rmse      13.789275
Name: 4499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        7.696800
val_mse      179.046700
val_r_sqr      0.947344
val_rmse      13.380833
Name: 4499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         195.611898
rmse         13.986132
mae           7.689333
r_sqr         0.945380
```

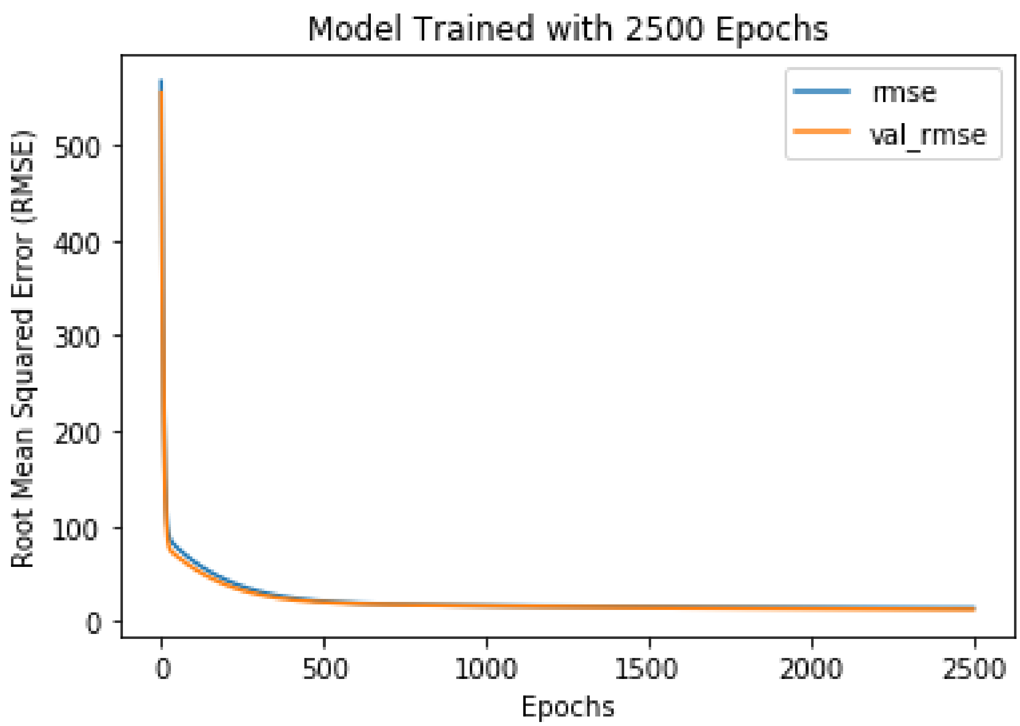
```
st_mse        0.001680
st_rmse       0.040985
st_mae        0.022904
st_r_sqr      0.855593
Name: 0, dtype: float64
================================================================================
====================



Model trained with 5000 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        9.628797
mse      317.097017
r_sqr      0.900590
rmse      17.807218
Name: 4999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae       10.011215
val_mse      323.438232
val_r_sqr      0.904837
val_rmse      17.984389
Name: 4999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         270.961722
rmse         16.460915
mae           9.025331
r_sqr         0.892656
st_mse        0.002518
st_rmse       0.050182
st_mae        0.027726
st_r_sqr      0.799610
Name: 0, dtype: float64
================================================================================
====================
```

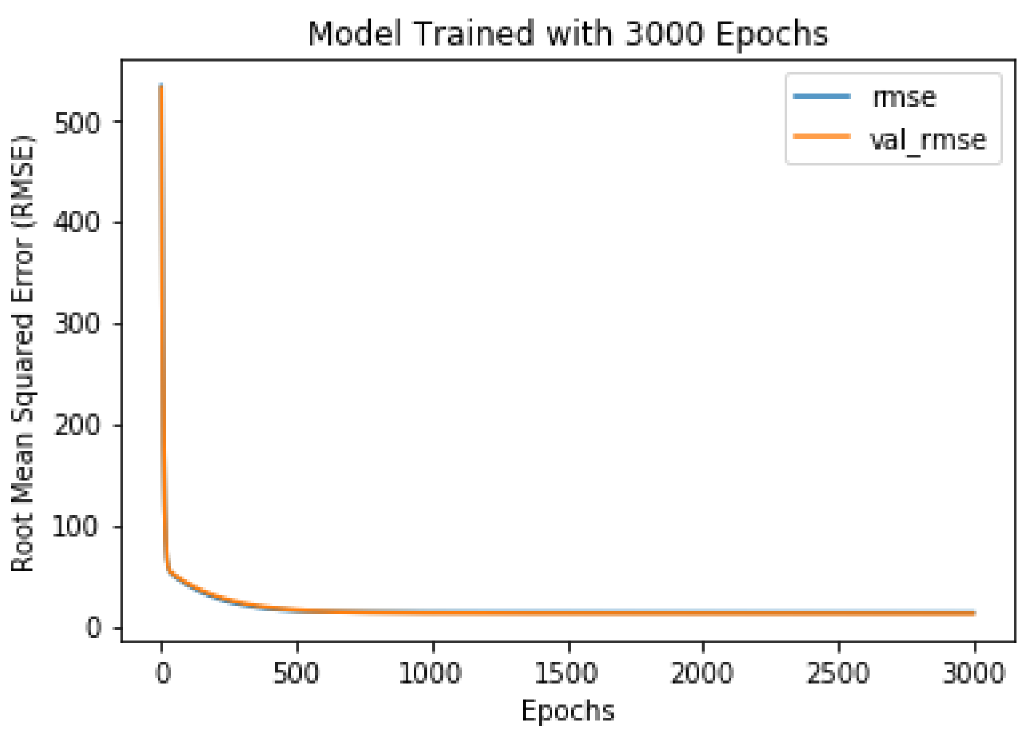

Model Trained with 500 Epochs
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
450
400
350
300
250
200
150
100
50
0
100
200
300
400
500

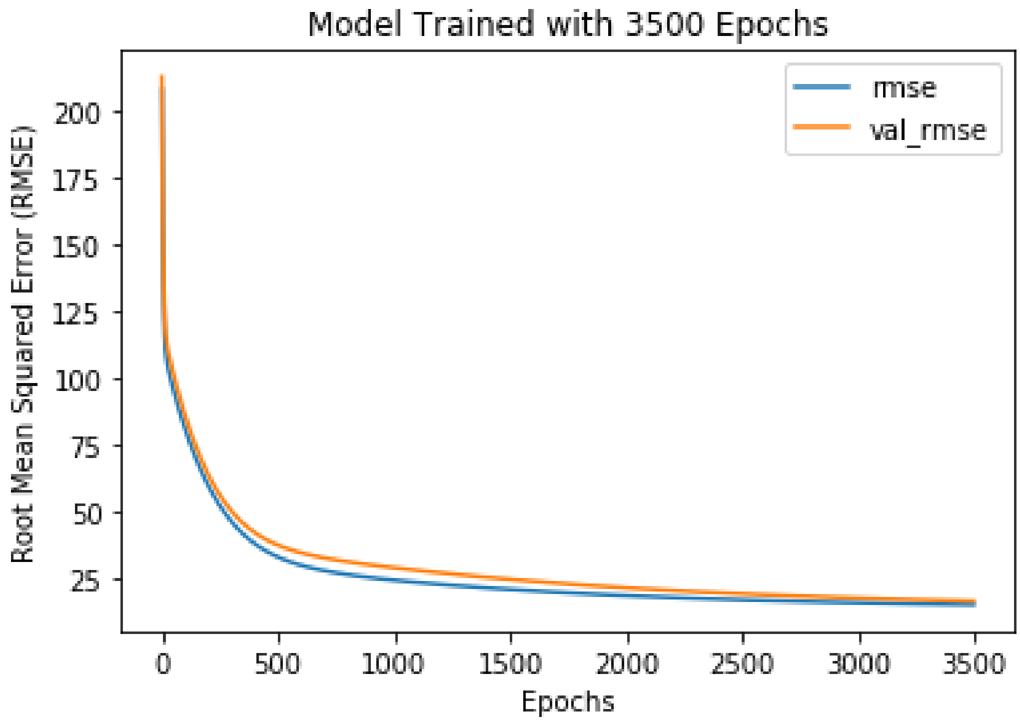

Model Trained with 1000 Epochs
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
250
200
150
100
50
0
200
400
600
800
1000

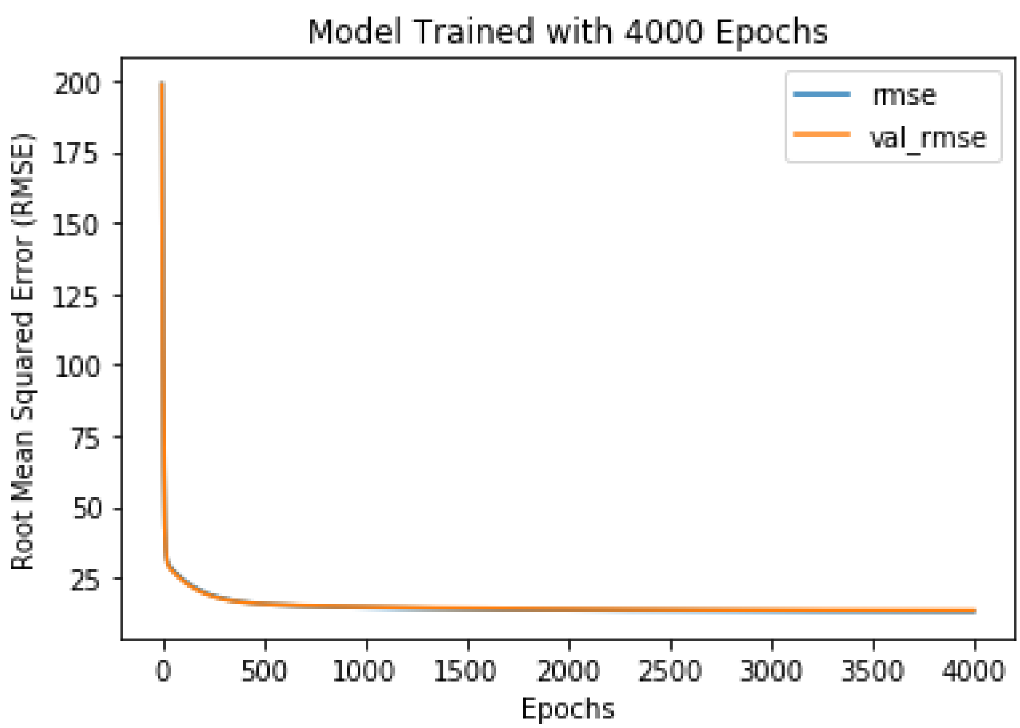

Model Trained with 1500 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
300
250
200
150
100
50
0
200
400
600
800
1000
1200
1400
Epochs

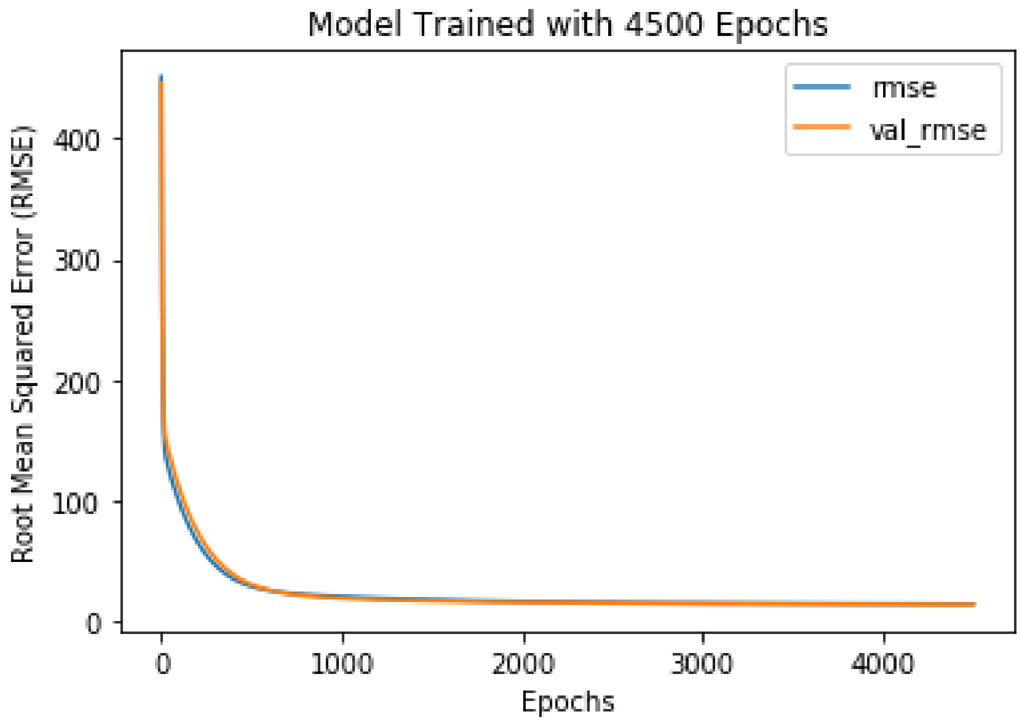

Model Trained with 2000 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
500
400
300
200
100
0
0
250
500
750
1000
1250
1500
1750
2000
Epochs

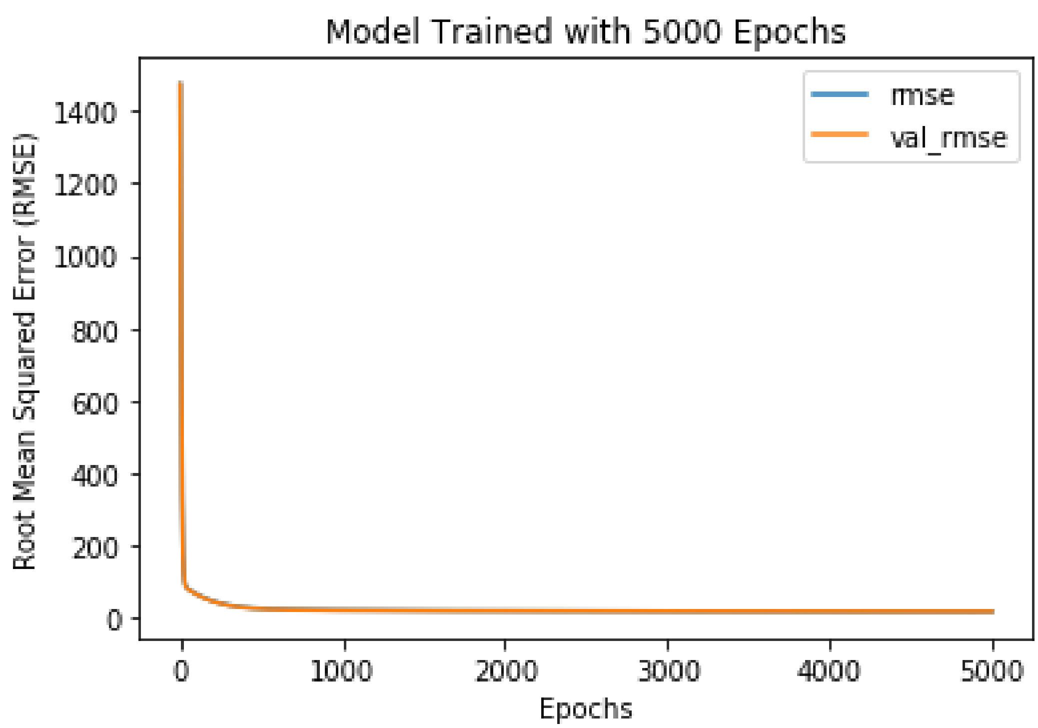

Model Trained with 2500 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
0
500
1000
1500
2000
2500
Epochs


Model Trained with 3000 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
0
500
1000
1500
2000
2500
3000
Epochs


Model Trained with 3500 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
200
175
150
125
100
75
50
25
0
500
1000
1500
2000
2500
3000
3500
Epochs


Model Trained with 4000 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
200
175
150
125
100
75
50
25
0
500
1000
1500
2000
2500
3000
3500
4000
Epochs


Model Trained with 4500 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
400
300
200
100
0
0
1000
2000
3000
4000
Epochs


Model Trained with 5000 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
1400
1200
1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
Epochs

- Models seem to overfit after about 800 to 1200 epochs
- Suitable number for training is around 1000

Learning Rate

[11]:
```
l_rate_tests = dict()
for i in range(1, 11):
    l_rate_tests[f"Test-{i}"] = build_train_test(
        fs,
        feature_cols,
        target_cols,
        layers=("auto", 1),
        activ_func="linear",
        epochs=1000,
        l_rate=0.1*i
    )
```

[12]:
```
for i in range(1,11):
    print(f"Model Trained with Learning Rate of {0.1*i}", end=f"\n{'-'*100}\n")

    print("Final Training results")
    print(l_rate_tests[f"Test-{i}"]["training_results"].iloc[-1, :4],␣
↪end=f"\n{'-'*100}\n")

    print("Final Validation results")
    print(l_rate_tests[f"Test-{i}"]["training_results"].iloc[-1, 8:12],␣
↪end=f"\n{'-'*100}\n")

    print("Test Set Results")
    print(l_rate_tests[f"Test-{i}"]["error_metrics"].iloc[0],␣
↪end=f"\n{'='*100}\n\n\n\n")

    ax = l_rate_tests[f"Test-{i}"]["training_results"].plot(
        y=["rmse", "val_rmse"], title=f"Model Trained with Learning Rate of␣
↪{i*0.1}",
    )
    ax.set_xlabel("Epochs")
    ax.set_ylabel("Root Mean Squared Error (RMSE)")
```

```
Model Trained with Learning Rate of 0.1
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        8.646140
mse      225.832283
r_sqr      0.932451
rmse      15.027717
Name: 999, dtype: float64
--------------------------------------------------------------------------------
```

```
--------------------
Final Validation results
val_mae        8.206920
val_mse      226.594284
val_r_sqr      0.911532
val_rmse      15.053049
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         499.063453
rmse         22.339728
mae          13.266776
r_sqr         0.827178
st_mse        0.001469
st_rmse       0.038327
st_mae        0.021745
st_r_sqr      0.920706
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.2
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        8.149623
mse      183.333082
r_sqr      0.936561
rmse      13.540055
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.724627
val_mse      259.048363
val_r_sqr      0.930949
val_rmse      16.094979
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         279.405541
rmse         16.715428
mae          10.350494
r_sqr         0.909043
```

```
st_mse        0.001193
st_rmse       0.034541
st_mae        0.019907
st_r_sqr      0.922097
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.30000000000000004
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        8.875316
mse      232.164985
r_sqr      0.916766
rmse      15.236961
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        9.317005
val_mse      264.603169
val_r_sqr      0.936175
val_rmse      16.266627
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         550.630922
rmse         23.465526
mae          13.402733
r_sqr         0.815074
st_mse        0.004784
st_rmse       0.069170
st_mae        0.037795
st_r_sqr      0.807651
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.4
--------------------------------------------------------------------------------
--------------------
Final Training results
```

```
mae        9.358519
mse      268.980714
r_sqr      0.909705
rmse      16.400632
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.922762
val_mse      244.808540
val_r_sqr      0.919457
val_rmse      15.646359
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         469.365558
rmse         21.664846
mae          12.242115
r_sqr         0.866366
st_mse        0.001745
st_rmse       0.041778
st_mae        0.021946
st_r_sqr      0.913637
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.5
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.682250
mse      136.031159
r_sqr      0.951552
rmse      11.663240
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        6.567995
val_mse      155.812608
val_r_sqr      0.956987
val_rmse      12.482492
Name: 999, dtype: float64
--------------------------------------------------------------------------------
```

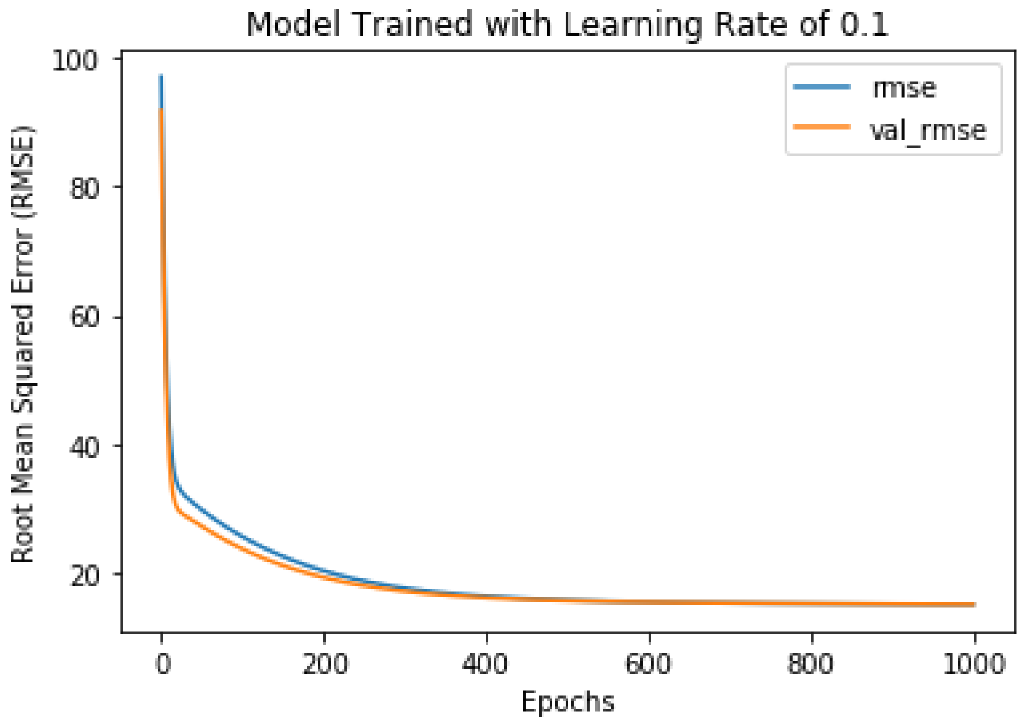
```
--------------------
Test Set Results
mse         204.102515
rmse         14.286445
mae           8.496443
r_sqr         0.940618
st_mse        0.001763
st_rmse       0.041987
st_mae        0.022318
st_r_sqr      0.896972
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.6000000000000001
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        8.614596
mse      221.458964
r_sqr      0.932705
rmse      14.881497
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.635414
val_mse      211.222346
val_r_sqr      0.924020
val_rmse      14.533490
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         268.766229
rmse         16.394091
mae           9.765973
r_sqr         0.905441
st_mse        0.000744
st_rmse       0.027267
st_mae        0.015875
st_r_sqr      0.947569
Name: 0, dtype: float64
================================================================================
====================
```

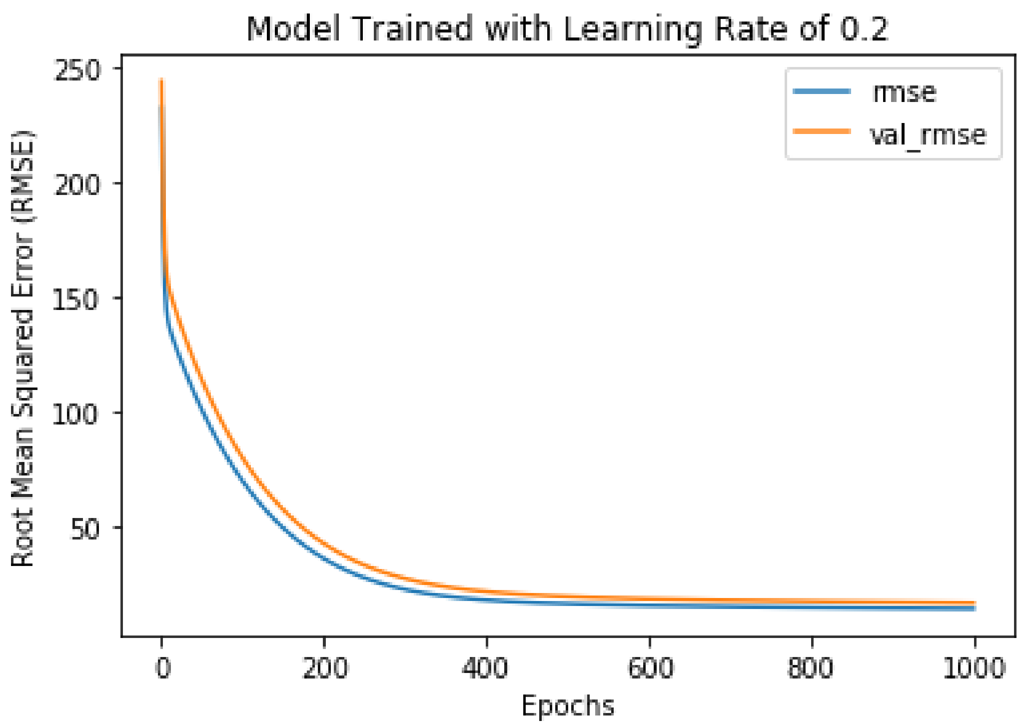
```
Model Trained with Learning Rate of 0.7000000000000001
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        5.532851
mse      117.335432
r_sqr      0.962092
rmse      10.832148
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        6.086643
val_mse      151.433658
val_r_sqr      0.955680
val_rmse      12.305838
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         2144.682506
rmse          46.310717
mae           27.081051
r_sqr          0.227825
st_mse         0.001620
st_rmse        0.040247
st_mae         0.021826
st_r_sqr       0.910596
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.8
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        5.954080
mse      127.160886
r_sqr      0.956463
rmse      11.276564
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
```

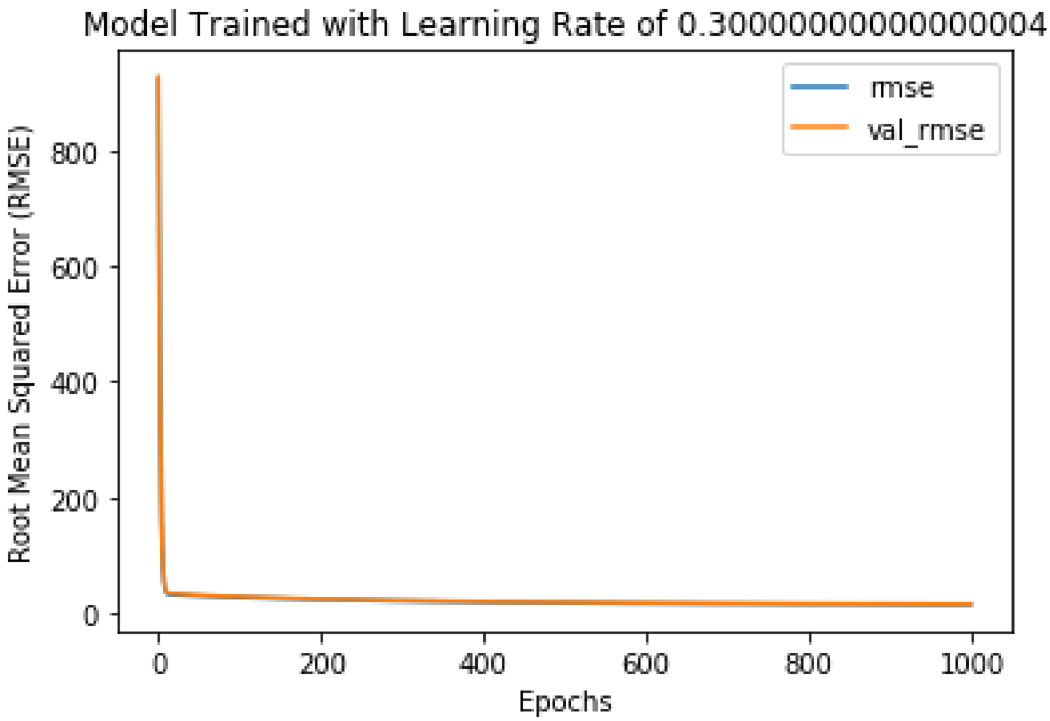
```
val_mae        7.456241
val_mse      202.591333
val_r_sqr      0.945969
val_rmse      14.233458
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         916.333690
rmse         30.271004
mae          18.496051
r_sqr         0.689704
st_mse        0.000795
st_rmse       0.028189
st_mae        0.015935
st_r_sqr      0.958190
Name: 0, dtype: float64
================================================================================
====================


Model Trained with Learning Rate of 0.9
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.346123
mse      135.253986
r_sqr      0.955358
rmse      11.629875
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        6.373476
val_mse      119.421225
val_r_sqr      0.956759
val_rmse      10.928002
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         328.133564
rmse         18.114457
mae          10.229089
r_sqr         0.909651
st_mse        0.000777
st_rmse       0.027875
```

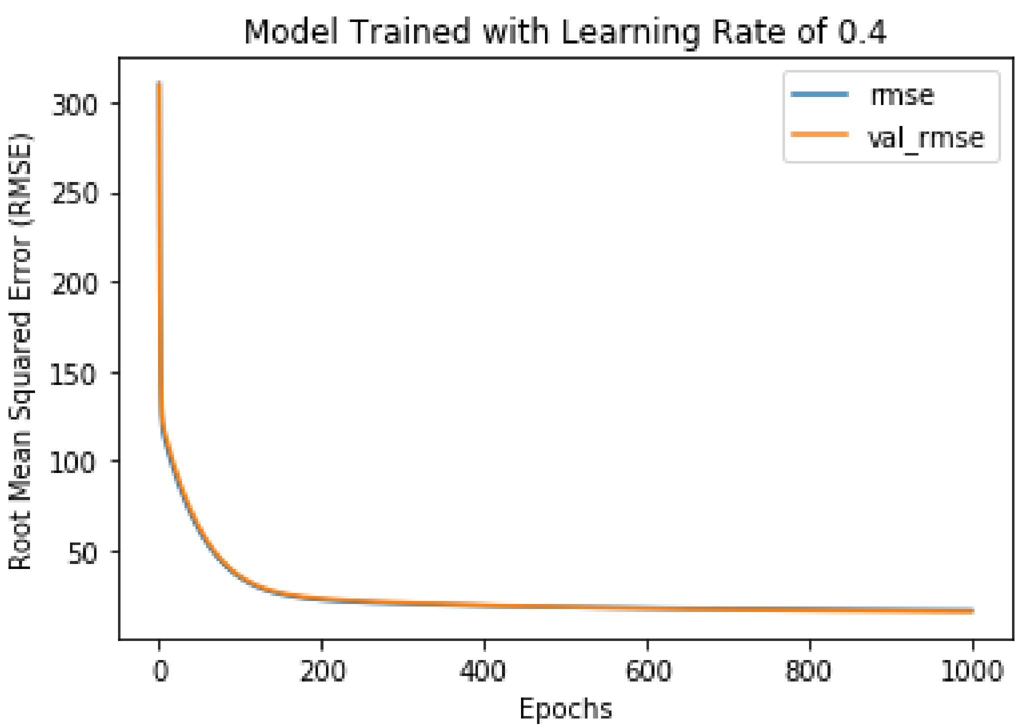
```
st_mae        0.014975
st_r_sqr      0.957077
Name: 0, dtype: float64
================================================================================
====================
```

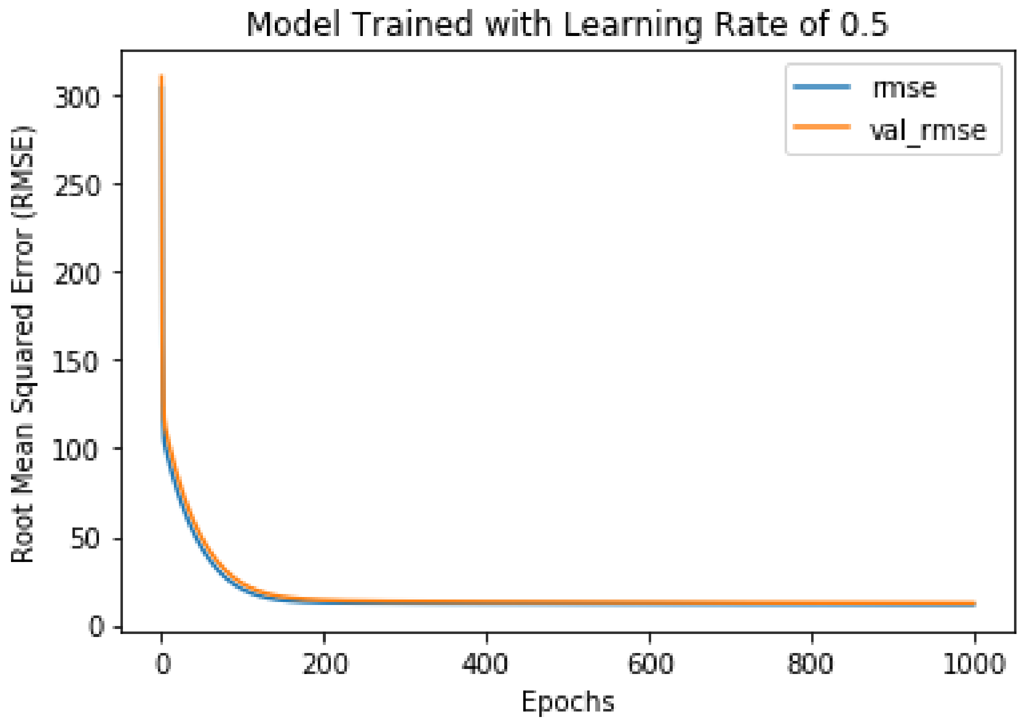
```
Model Trained with Learning Rate of 1.0
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        7.569875
mse      191.786131
r_sqr      0.938449
rmse      13.848687
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        7.170502
val_mse      190.116785
val_r_sqr      0.933657
val_rmse      13.788284
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         581.593532
rmse         24.116250
mae          14.473857
r_sqr         0.822551
st_mse        0.000960
st_rmse       0.030981
st_mae        0.016611
st_r_sqr      0.949105
Name: 0, dtype: float64
================================================================================
====================
```

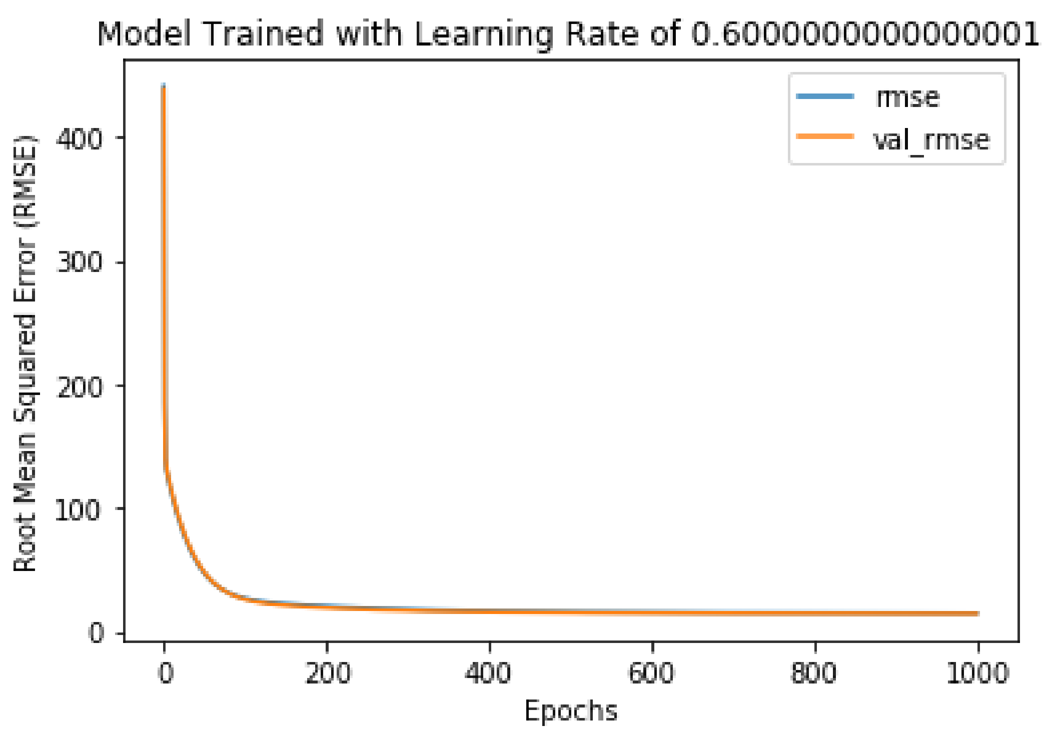

Model Trained with Learning Rate of 0.1
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
100
80
60
40
20
0
200
400
600
800
1000

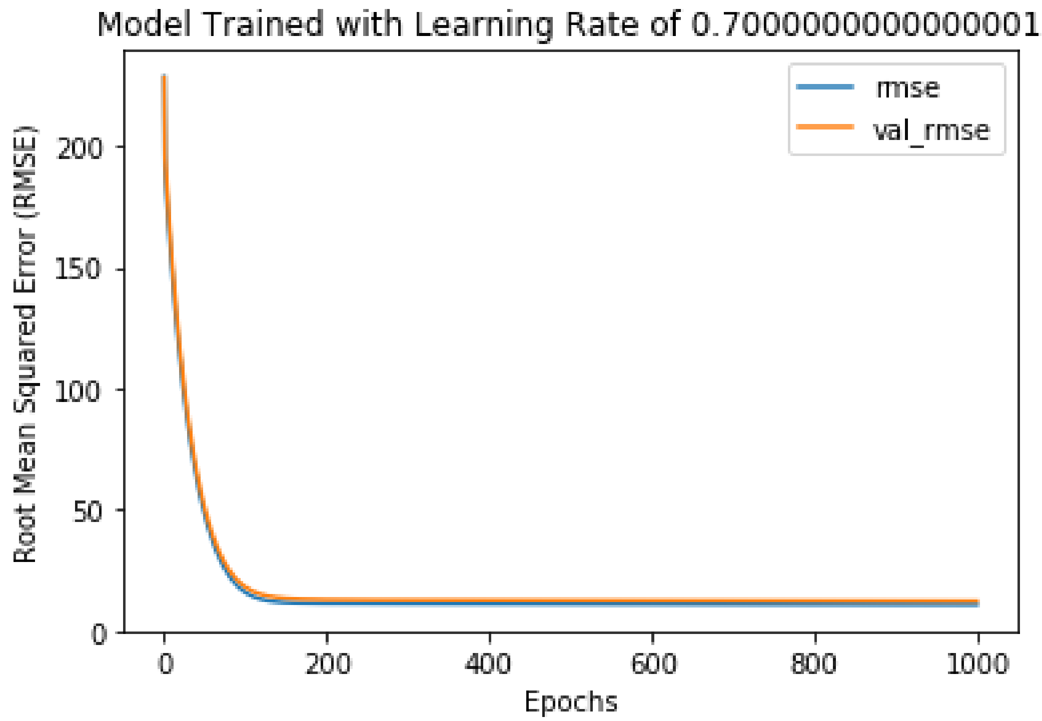

Model Trained with Learning Rate of 0.2
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
250
200
150
100
50
0
200
400
600
800
1000

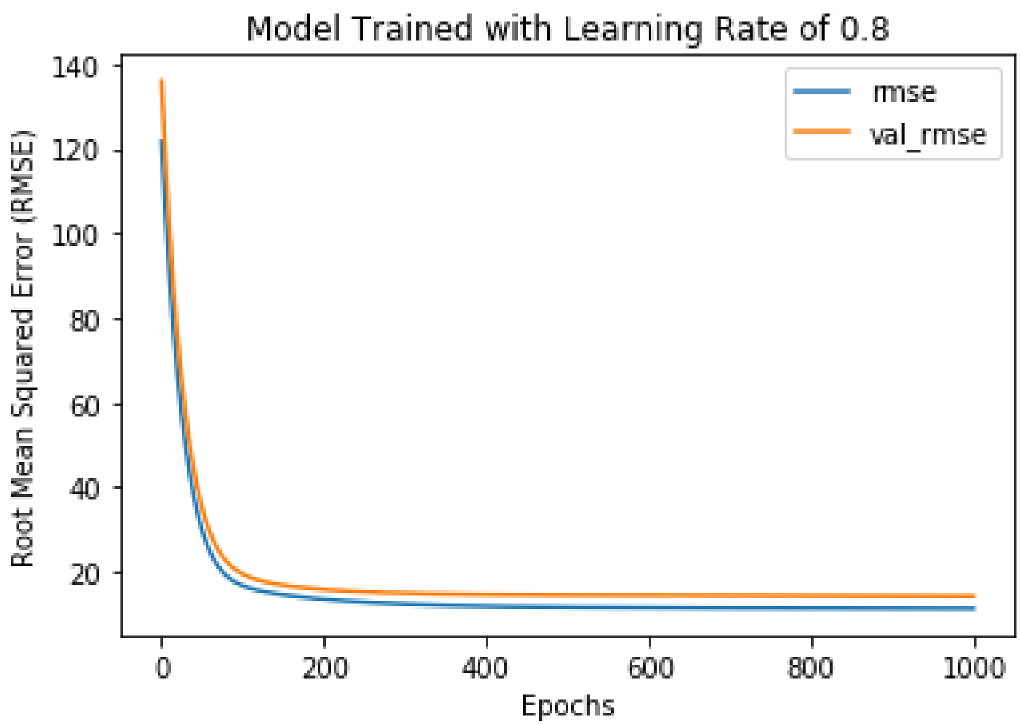

Model Trained with Learning Rate of 0.30000000000000004
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
200
400
600
800
1000
Epochs

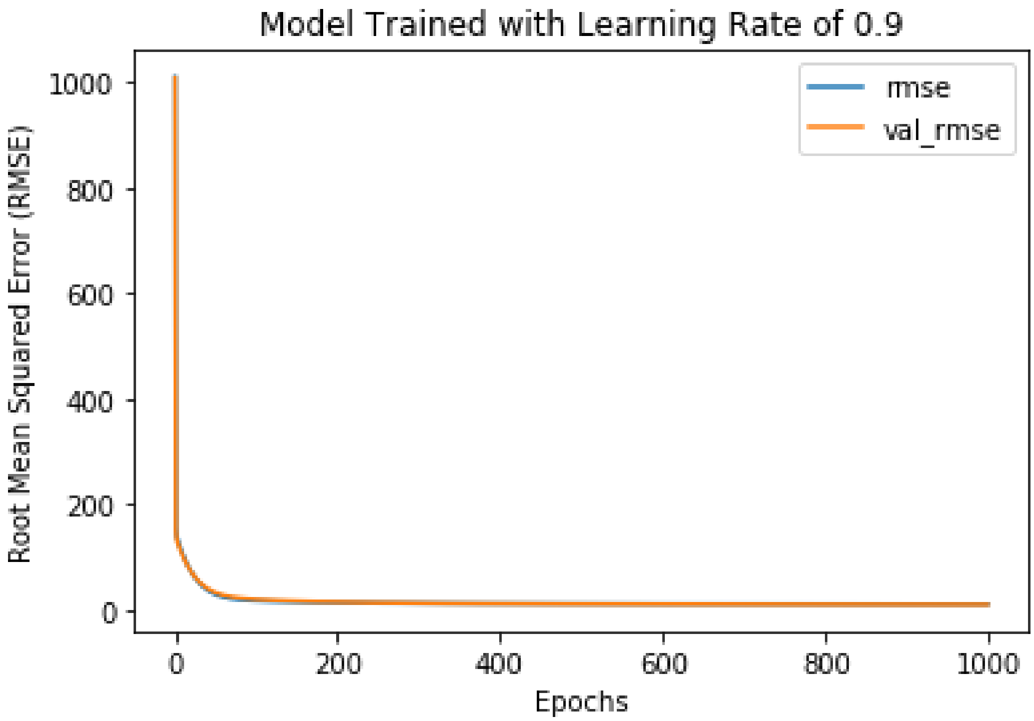

Model Trained with Learning Rate of 0.4
rmse
val_rmse
Root Mean Squared Error (RMSE)
50
100
150
200
250
300
0
200
400
600
800
1000
Epochs

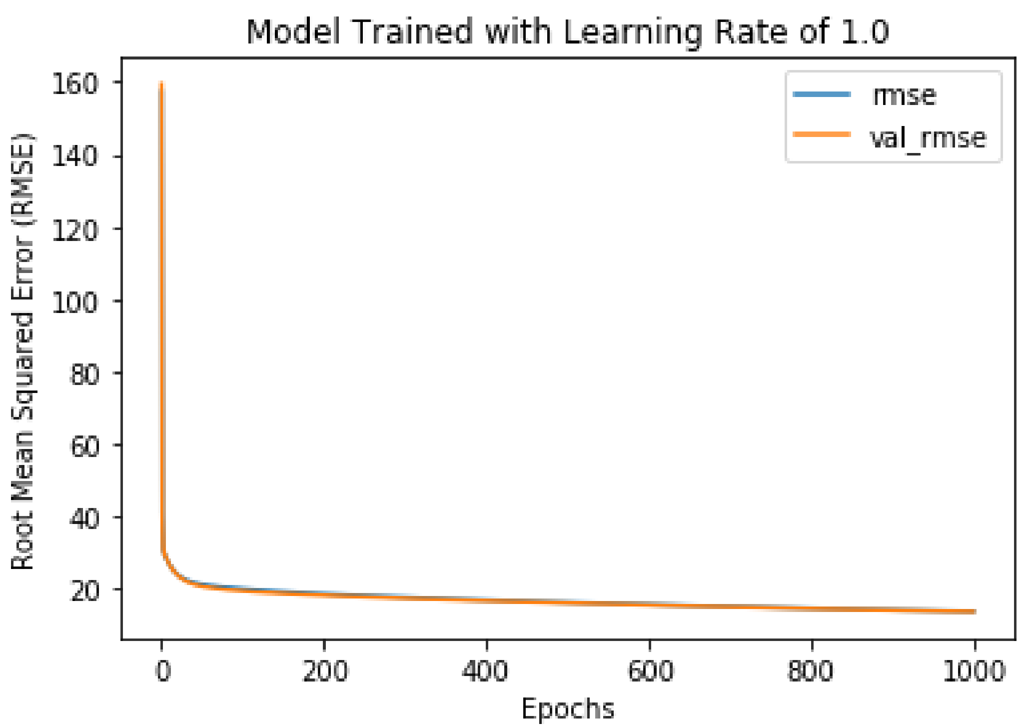

Model Trained with Learning Rate of 0.5
rmse
val_rmse
Root Mean Squared Error (RMSE)
300
250
200
150
100
50
0
0
200
400
600
800
1000
Epochs


Model Trained with Learning Rate of 0.6000000000000001
rmse
val_rmse
Root Mean Squared Error (RMSE)
400
300
200
100
0
0
200
400
600
800
1000
Epochs


Model Trained with Learning Rate of 0.7000000000000001
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
50
100
150
200
0
200
400
600
800
1000
Epochs


Model Trained with Learning Rate of 0.8
rmse
val_rmse
Root Mean Squared Error (RMSE)
20
40
60
80
100
120
140
0
200
400
600
800
1000
Epochs


Model Trained with Learning Rate of 0.9
rmse
val_rmse
Root Mean Squared Error (RMSE)
1000
800
600
400
200
0
0
200
400
600
800
1000
Epochs


Model Trained with Learning Rate of 1.0
rmse
val_rmse
Root Mean Squared Error (RMSE)
160
140
120
100
80
60
40
20
0
200
400
600
800
1000
Epochs

- learning rate of 0.3 appears to offer most consistent resuls between training, validation and testing sets

Activation Functions

[13]:
```
activ_tests = dict()
activations = ("sigmoid", "tanh", "relu", "linear")

for func in activations:
    activ_tests[func] = build_train_test(
        fs,
        feature_cols,
        target_cols,
        layers=("auto", 1),
        activ_func=func,
        epochs=1000,
        l_rate=0.3
    )
```

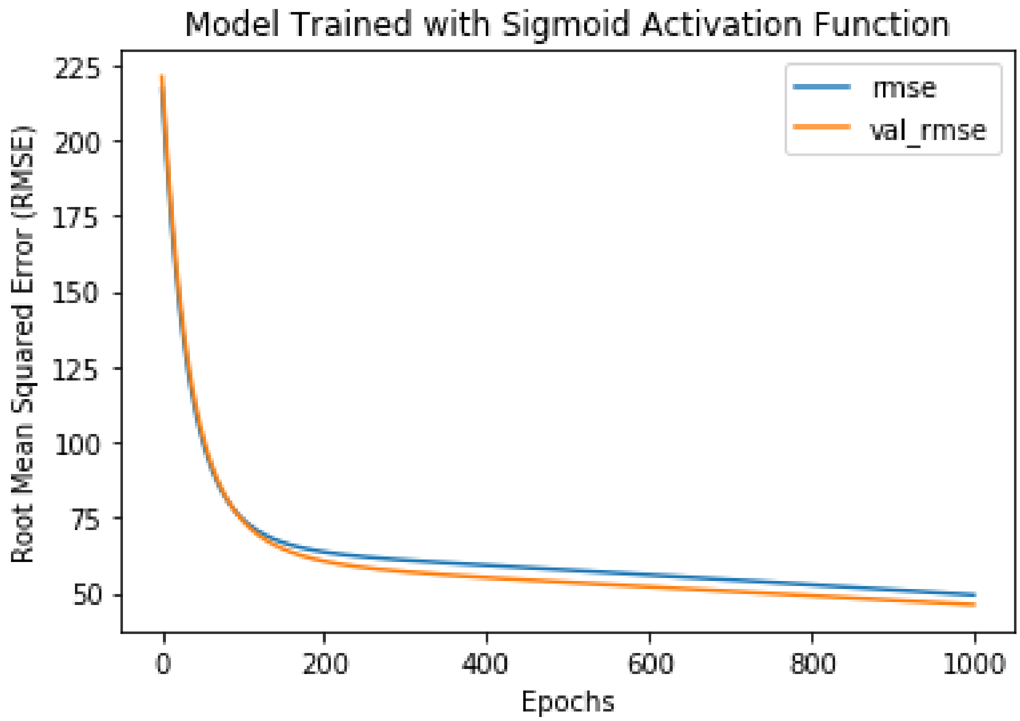
[14]:
```
for a in activations:
    print(f"Model Trained with {a.capitalize()} Activation Function",␣
↪end=f"\n{'-'*100}\n")

    print("Final Training Results")
    print(activ_tests[a]["training_results"].iloc[-1, :4], end=f"\n{'-'*100}\n")

    print("Final Validation Results")
    print(activ_tests[a]["training_results"].iloc[-1, 8:12],␣
↪end=f"\n{'-'*100}\n")

    print("Test Set Results")
    print(activ_tests[a]["error_metrics"].iloc[0], end=f"\n{'='*100}\n\n\n\n")

    ax = activ_tests[a]["training_results"].plot(
        y=["rmse", "val_rmse"], title=f"Model Trained with {a.capitalize()}␣
↪Activation Function",
    )
    ax.set_xlabel("Epochs")
    ax.set_ylabel("Root Mean Squared Error (RMSE)")
```

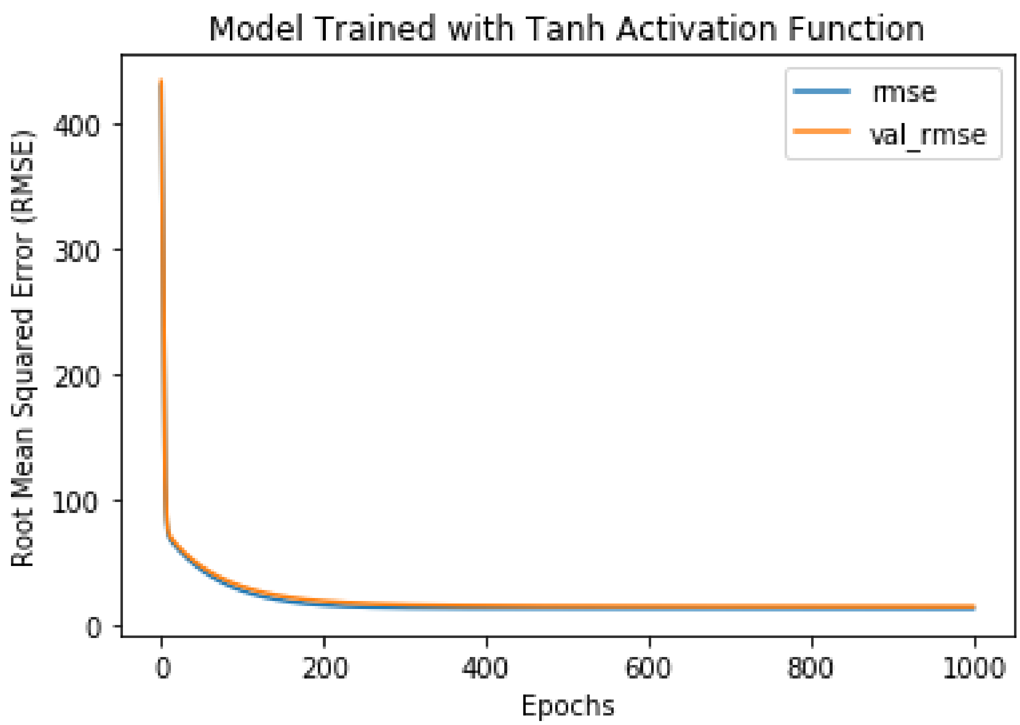
```
Model Trained with Sigmoid Activation Function
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae         37.054013
mse       2444.546711
r_sqr        0.245836
rmse        49.442357
Name: 999, dtype: float64
```

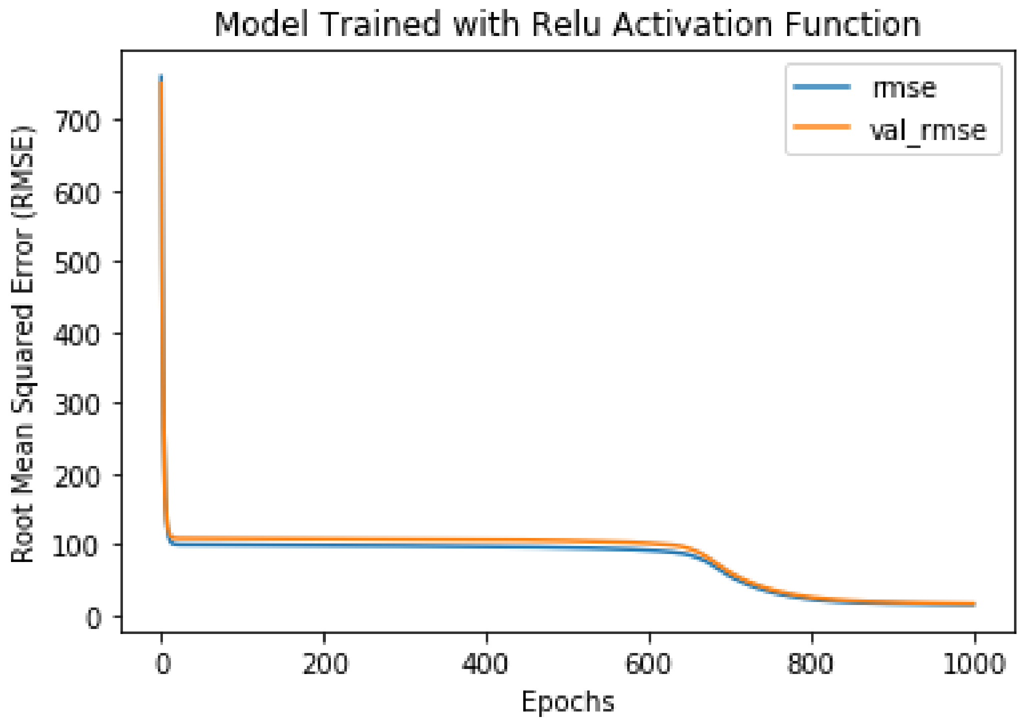
```
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        36.524513
val_mse      2135.781825
val_r_sqr       0.215808
val_rmse       46.214520
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         2216.249955
rmse          47.077064
mae           36.780372
r_sqr          0.267296
st_mse         0.011574
st_rmse        0.107584
st_mae         0.075433
st_r_sqr       0.231381
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Tanh Activation Function
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.083939
mse      161.562093
r_sqr      0.946840
rmse      12.710708
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.491849
val_mse      206.925613
val_r_sqr      0.942838
val_rmse      14.384909
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1363.293353
rmse          36.922803
mae           24.791772
```

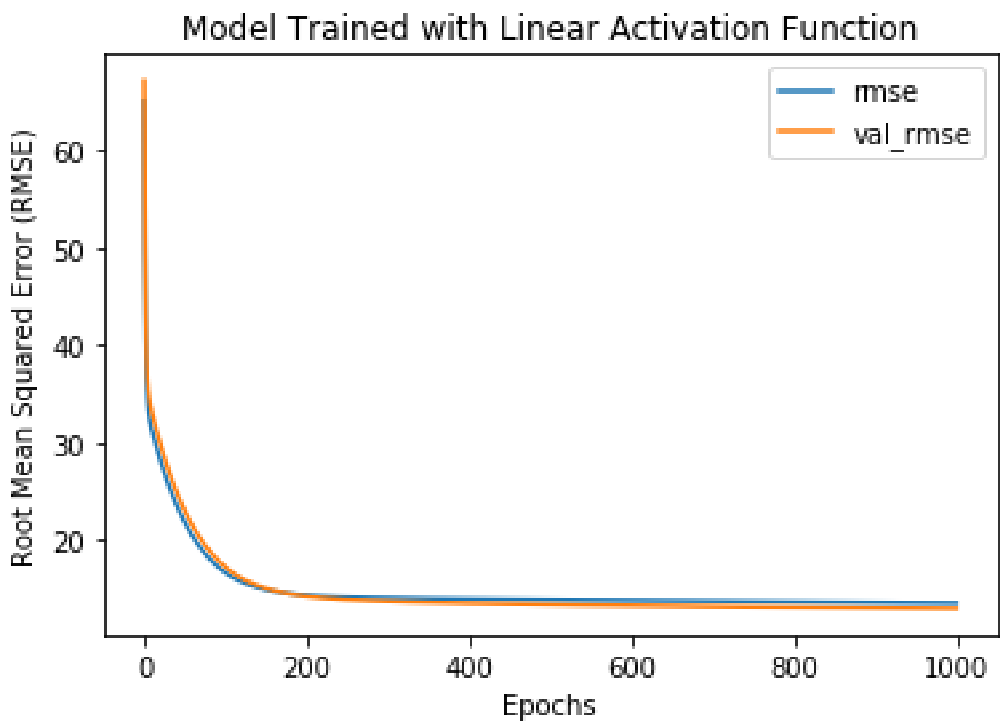
```
r_sqr          0.504479
st_mse         0.001226
st_rmse        0.035017
st_mae         0.020053
st_r_sqr       0.935344
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Relu Activation Function
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.207435
mse      212.462493
r_sqr      0.930997
rmse      14.576093
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.290627
val_mse      254.122270
val_r_sqr      0.934525
val_rmse      15.941213
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1365.702469
rmse          36.955412
mae           24.426133
r_sqr          0.420647
st_mse         0.000955
st_rmse        0.030899
st_mae         0.019378
st_r_sqr       0.937721
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Linear Activation Function
--------------------------------------------------------------------------------
--------------------
```

```
Final Training Results
mae        7.397667
mse      183.284561
r_sqr      0.941414
rmse      13.538263
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        6.809972
val_mse      168.663900
val_r_sqr      0.952675
val_rmse      12.987067
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1335.219034
rmse          36.540649
mae           22.382932
r_sqr          0.472678
st_mse         0.001089
st_rmse        0.032999
st_mae         0.018397
st_r_sqr       0.937636
Name: 0, dtype: float64
================================================================================
====================
```


Model Trained with Sigmoid Activation Function
Root Mean Squared Error (RMSE)
225
200
175
150
125
100
75
50
0
200
400
600
800
1000
Epochs
rmse
val_rmse


Model Trained with Tanh Activation Function
Root Mean Squared Error (RMSE)
400
300
200
100
0
0
200
400
600
800
1000
Epochs
rmse
val_rmse


Model Trained with Relu Activation Function
rmse
val_rmse
Root Mean Squared Error (RMSE)
700
600
500
400
300
200
100
0
0
200
400
600
800
1000
Epochs


Model Trained with Linear Activation Function
rmse
val_rmse
Root Mean Squared Error (RMSE)
60
50
40
30
20
0
200
400
600
800
1000
Epochs

- tanh is quite clearly the best performing activation function

Hidden Layers

[15]:
```
layer_tests = dict()
for i in range(1, 11):
    layer_tests[f"Test-{i}"] = build_train_test(
        fs,
        feature_cols,
        target_cols,
        layers=("auto", i, 1),
        activ_func="tanh",
        epochs=1000,
        l_rate=0.3
    )
```

[16]:
```
for i in range(1,11):
    print(f"Model trained with Single Hidden Layer of Size {i}",␣
↪end=f"\n{'-'*100}\n")

    print("Final Training Results")
    print(layer_tests[f"Test-{i}"]["training_results"].iloc[-1, :4],␣
↪end=f"\n{'-'*100}\n")

    print("Final Validation Results")
    print(layer_tests[f"Test-{i}"]["training_results"].iloc[-1, 8:12],␣
↪end=f"\n{'-'*100}\n")

    print("Test Set Results")
    print(layer_tests[f"Test-{i}"]["error_metrics"].iloc[0],␣
↪end=f"\n{'='*100}\n\n\n\n")

    ax = layer_tests[f"Test-{i}"]["training_results"].plot(
        y=["rmse", "val_rmse"], title=f"Model trained with Single Hidden Layer␣
↪of Size {i}",
    )
    ax.set_xlabel("Epochs")
    ax.set_ylabel("Root Mean Squared Error (RMSE)")
```

```
Model trained with Single Hidden Layer of Size 1
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.064012
mse      192.208285
r_sqr      0.941583
rmse      13.863920
```

```
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.717680
val_mse      171.990340
val_r_sqr      0.943096
val_rmse      13.114509
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         847.342010
rmse         29.109140
mae          19.468714
r_sqr         0.671788
st_mse        0.001094
st_rmse       0.033083
st_mae        0.018382
st_r_sqr      0.944212
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 2
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.915232
mse      220.109533
r_sqr      0.927152
rmse      14.836089
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.058116
val_mse      215.478264
val_r_sqr      0.942478
val_rmse      14.679178
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         233.035666
rmse         15.265506
```

```
mae          10.703701
r_sqr         0.912676
st_mse        0.001095
st_rmse       0.033083
st_mae        0.017417
st_r_sqr      0.917615
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 3
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        9.254691
mse      270.616416
r_sqr      0.905137
rmse      16.450423
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.867702
val_mse      257.389892
val_r_sqr      0.928529
val_rmse      16.043375
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1078.781547
rmse          32.844810
mae           22.570404
r_sqr          0.671273
st_mse         0.002443
st_rmse        0.049426
st_mae         0.025979
st_r_sqr       0.883922
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 4
--------------------------------------------------------------------------------
```

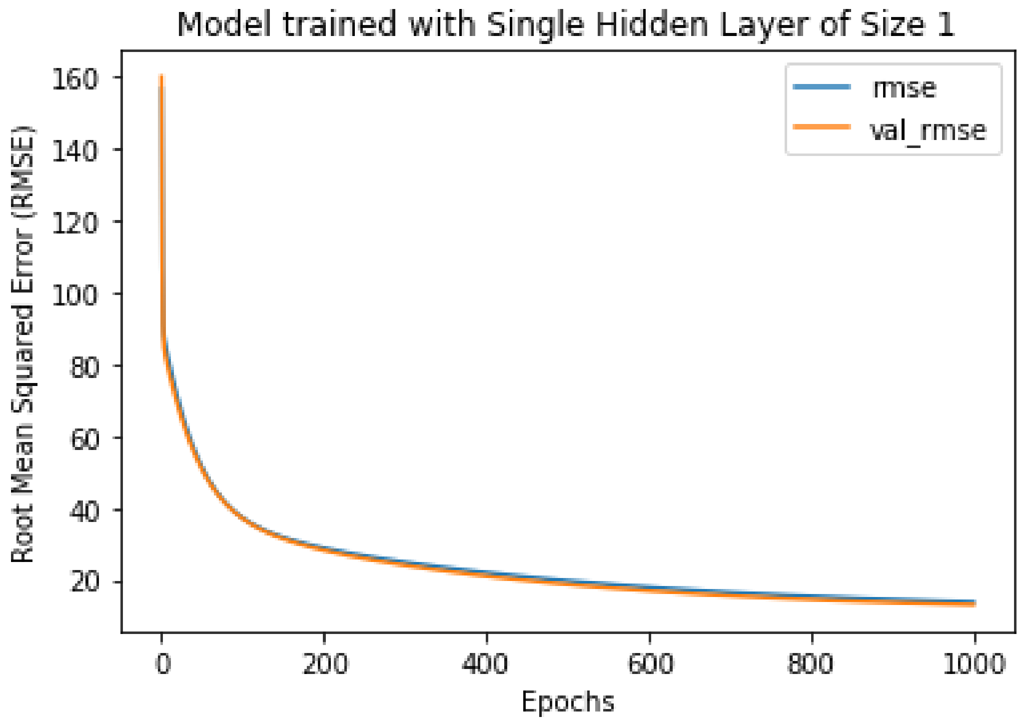
```
--------------------
Final Training Results
mae        8.600294
mse      222.283558
r_sqr      0.929792
rmse      14.909177
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.790956
val_mse      229.241784
val_r_sqr      0.919376
val_rmse      15.140733
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1726.179383
rmse          41.547315
mae           27.559650
r_sqr          0.452181
st_mse         0.001560
st_rmse        0.039501
st_mae         0.023094
st_r_sqr       0.922811
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 5
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae       11.540708
mse      368.044396
r_sqr      0.870646
rmse      19.184483
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae       14.414949
val_mse      672.346771
val_r_sqr      0.827337
val_rmse      25.929650
```

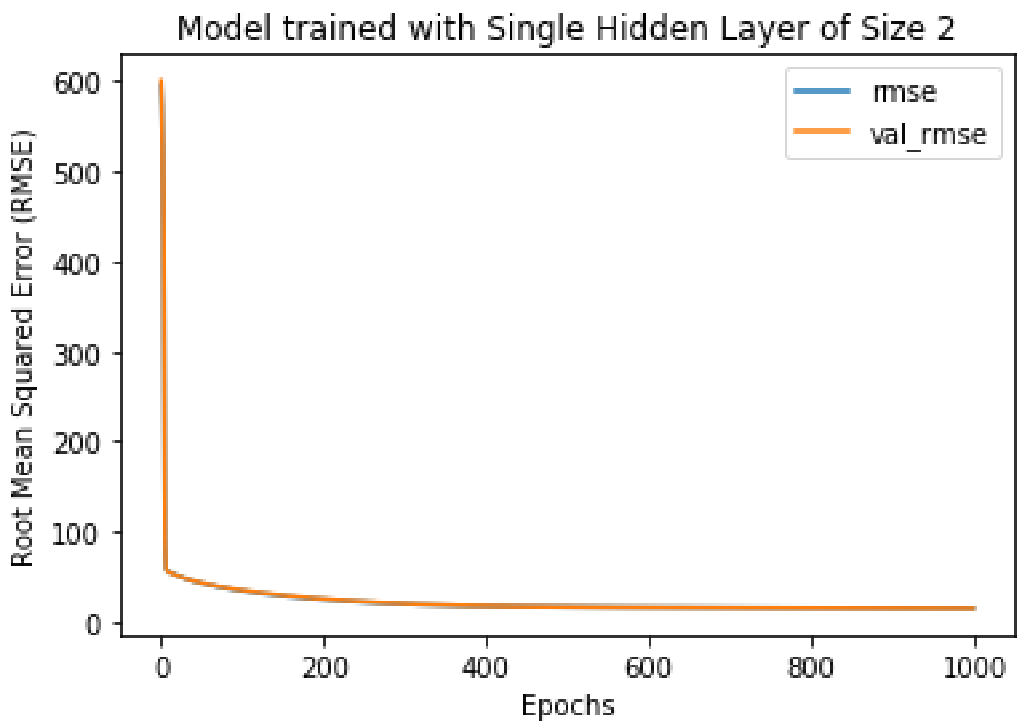
```
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         528.112732
rmse         22.980703
mae          14.177557
r_sqr         0.826392
st_mse        0.008521
st_rmse       0.092310
st_mae        0.051561
st_r_sqr      0.664568
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 6
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.790562
mse      238.483448
r_sqr      0.916710
rmse      15.442909
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.148302
val_mse      263.229903
val_r_sqr      0.922821
val_rmse      16.224361
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         537.106556
rmse         23.175559
mae          13.916583
r_sqr         0.845319
st_mse        0.002910
st_rmse       0.053947
st_mae        0.026422
st_r_sqr      0.871343
Name: 0, dtype: float64
================================================================================
```

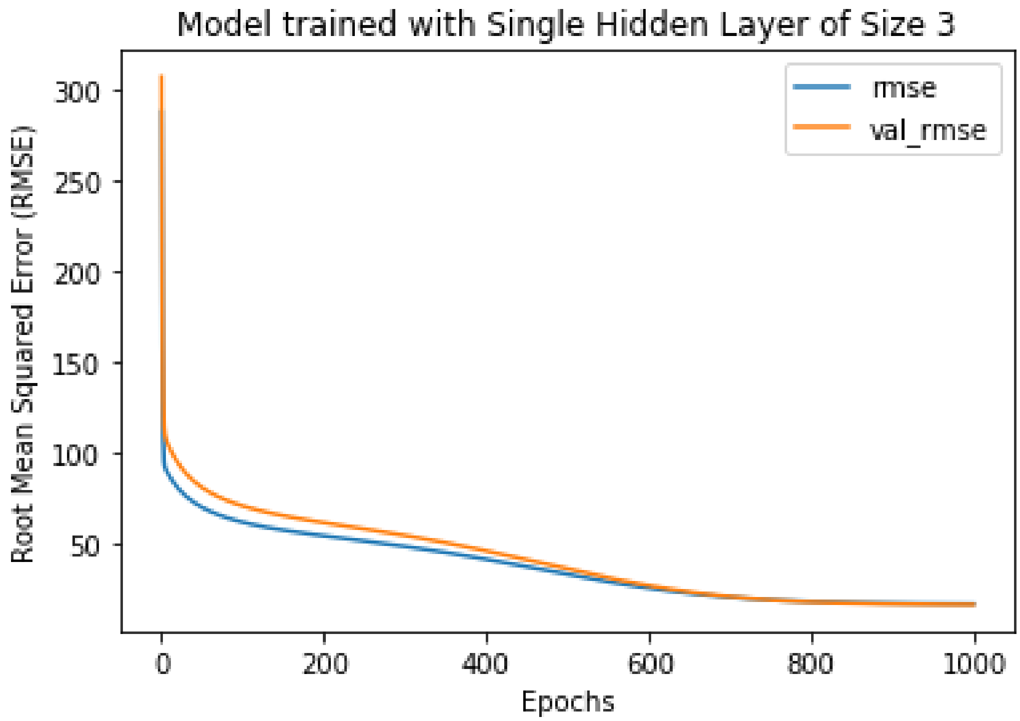
```
====================


Model trained with Single Hidden Layer of Size 7
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        9.686895
mse      282.799530
r_sqr      0.916667
rmse      16.816644
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.006291
val_mse      248.341253
val_r_sqr      0.910603
val_rmse      15.758847
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         478.347390
rmse         21.871154
mae          13.226941
r_sqr         0.809252
st_mse        0.001348
st_rmse       0.036720
st_mae        0.019616
st_r_sqr      0.906809
Name: 0, dtype: float64
================================================================================
====================


Model trained with Single Hidden Layer of Size 8
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.573392
mse      212.627106
r_sqr      0.930075
rmse      14.581739
Name: 999, dtype: float64
--------------------------------------------------------------------------------
```

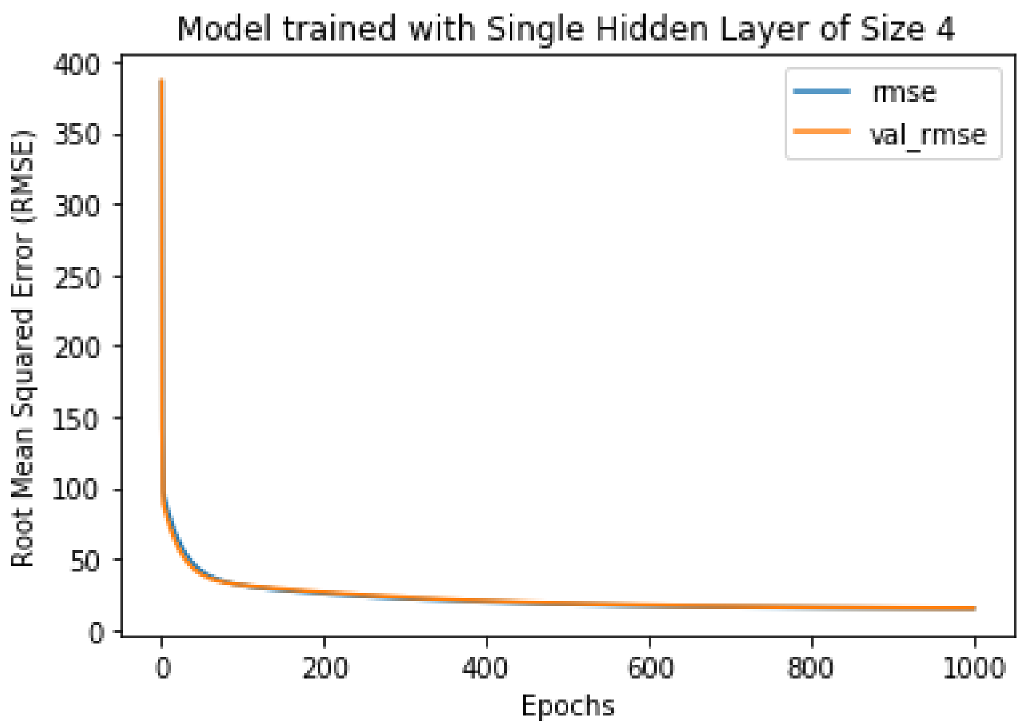
```
--------------------
Final Validation Results
val_mae        8.780433
val_mse      240.458554
val_r_sqr      0.931281
val_rmse      15.506726
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         364.947716
rmse         19.103605
mae          10.540391
r_sqr         0.872627
st_mse        0.000673
st_rmse       0.025940
st_mae        0.016634
st_r_sqr      0.927550
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 9
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae       10.862719
mse      283.080187
r_sqr      0.915056
rmse      16.824987
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae       10.174926
val_mse      228.511120
val_r_sqr      0.903925
val_rmse      15.116584
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1065.580731
rmse          32.643234
mae           22.909868
r_sqr          0.655941
```

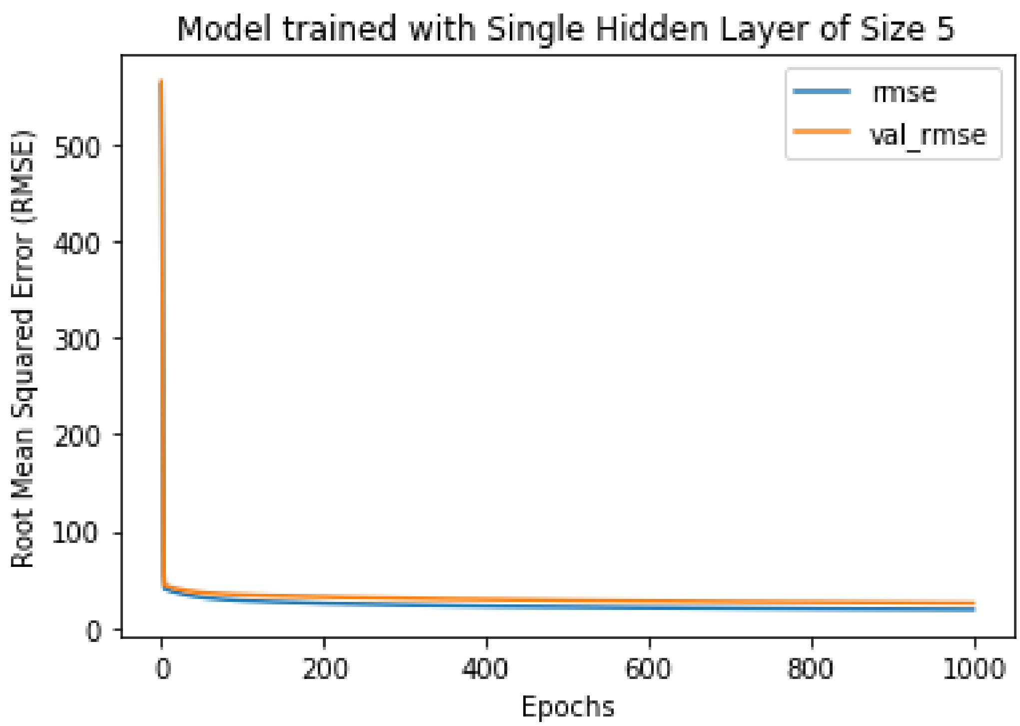
```
st_mse         0.002602
st_rmse        0.051005
st_mae         0.028064
st_r_sqr       0.889588
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 10
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.434438
mse      182.382977
r_sqr      0.944810
rmse      13.504924
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.809503
val_mse      265.871295
val_r_sqr      0.901530
val_rmse      16.305560
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1304.304852
rmse          36.115161
mae           24.295704
r_sqr          0.547036
st_mse         0.001535
st_rmse        0.039184
st_mae         0.023707
st_r_sqr       0.907362
Name: 0, dtype: float64
================================================================================
====================
```

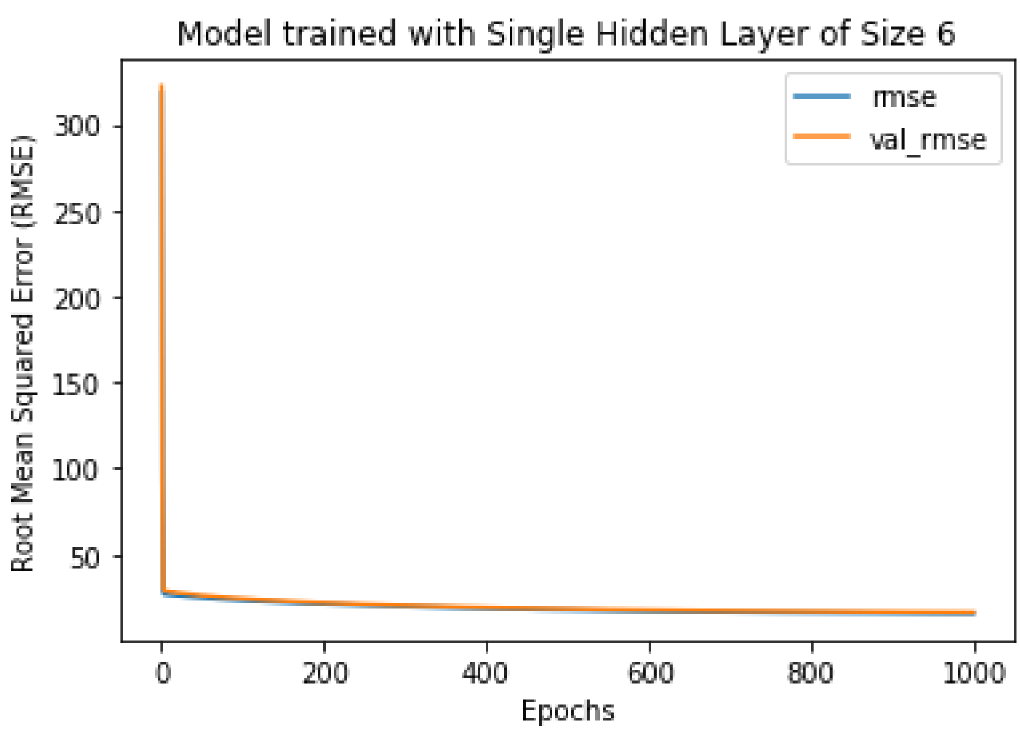

Model trained with Single Hidden Layer of Size 1
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
160
140
120
100
80
60
40
20
0
200
400
600
800
1000

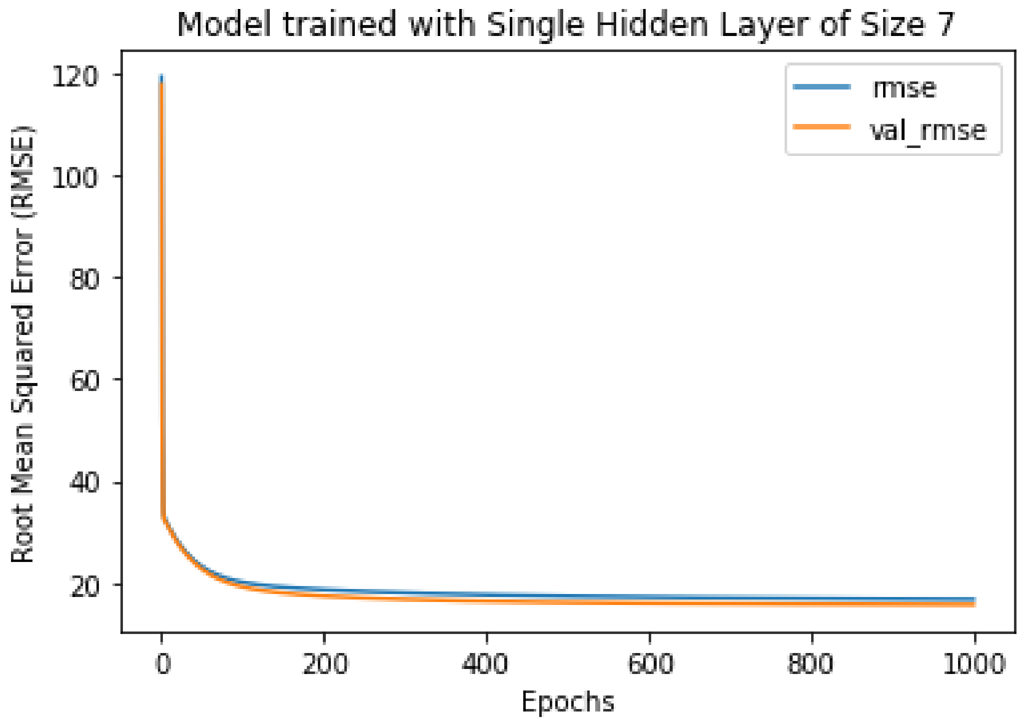

Model trained with Single Hidden Layer of Size 2
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
600
500
400
300
200
100
0
0
200
400
600
800
1000

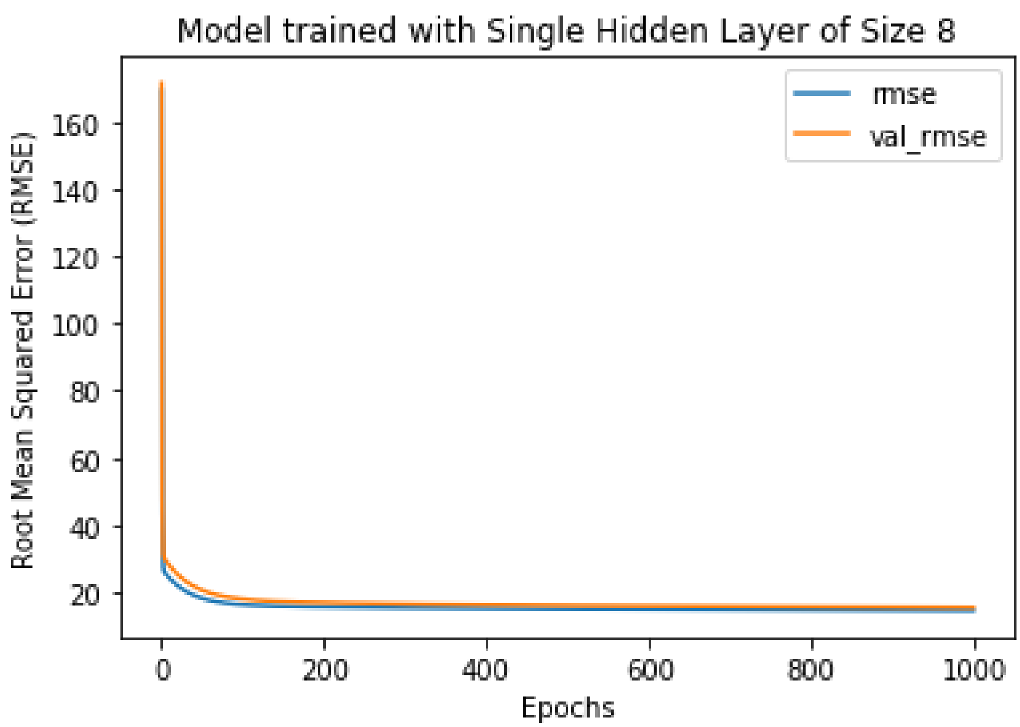

Model trained with Single Hidden Layer of Size 3
rmse
val_rmse
Root Mean Squared Error (RMSE)
300
250
200
150
100
50
0
200
400
600
800
1000
Epochs

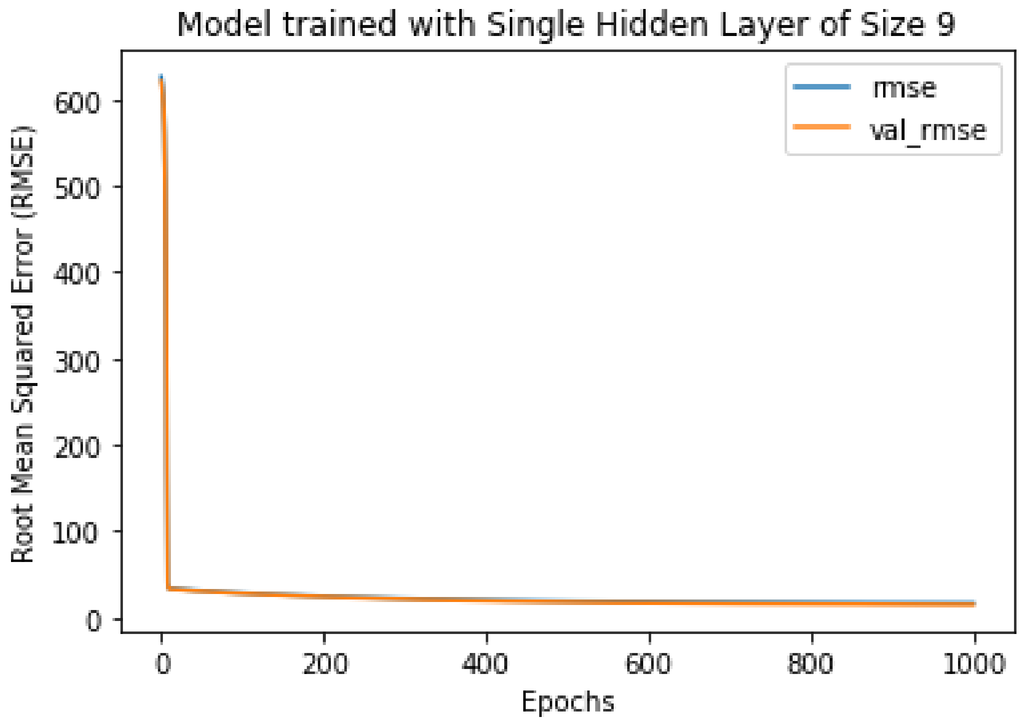

Model trained with Single Hidden Layer of Size 4
rmse
val_rmse
Root Mean Squared Error (RMSE)
400
350
300
250
200
150
100
50
0
0
200
400
600
800
1000
Epochs

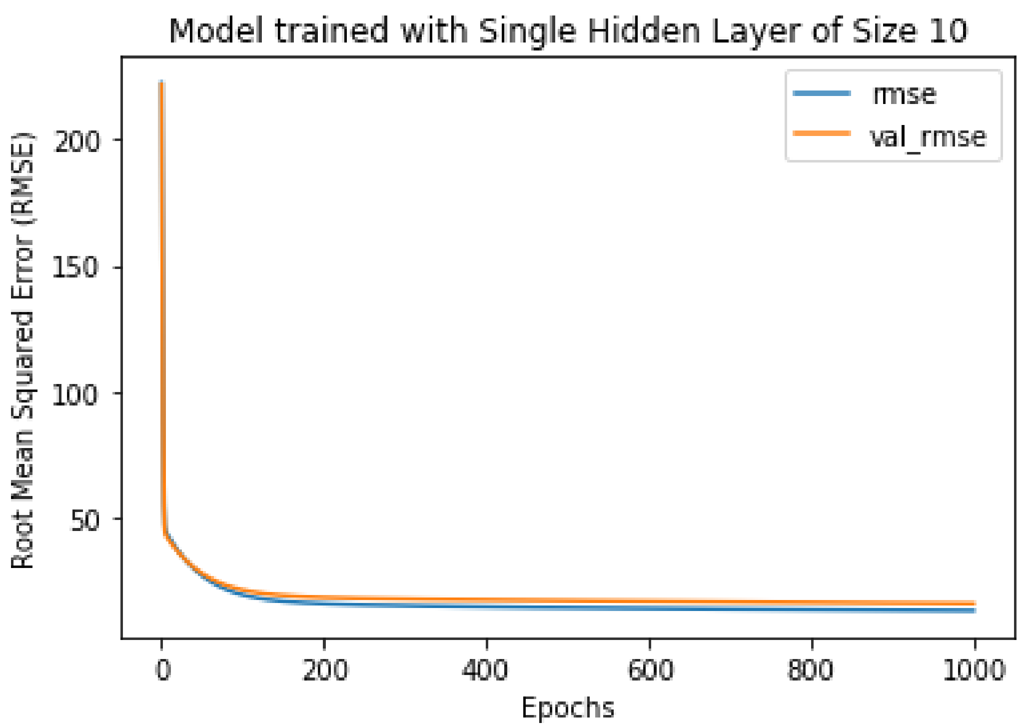

Model trained with Single Hidden Layer of Size 5
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
0
200
400
600
800
1000
Epochs


Model trained with Single Hidden Layer of Size 6
rmse
val_rmse
Root Mean Squared Error (RMSE)
50
100
150
200
250
300
0
200
400
600
800
1000
Epochs


Model trained with Single Hidden Layer of Size 7
rmse
val_rmse
Root Mean Squared Error (RMSE)
120
100
80
60
40
20
0
200
400
600
800
1000
Epochs


Model trained with Single Hidden Layer of Size 8
rmse
val_rmse
Root Mean Squared Error (RMSE)
160
140
120
100
80
60
40
20
0
200
400
600
800
1000
Epochs


Model trained with Single Hidden Layer of Size 9
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
600
0
200
400
600
800
1000
Epochs


Model trained with Single Hidden Layer of Size 10
rmse
val_rmse
Root Mean Squared Error (RMSE)
50
100
150
200
0
200
400
600
800
1000
Epochs

[17]:
```
layer_tests_2 = dict()
for i in range(1, 11):
    layer_tests_2[f"Test-{i}"] = build_train_test(
        fs,
        feature_cols,
        target_cols,
        layers=("auto", 10, i, 1),
        activ_func="tanh",
        epochs=1000,
        l_rate=0.3
    )
```

[18]:
```
for i in range(1,11):
    print(f"Model Trained with Hidden Layer Configuration (10, {i})",␣
↪end=f"\n{'-'*100}\n")

    print("Final Training Results")
    print(layer_tests_2[f"Test-{i}"]["training_results"].iloc[-1, :4],␣
↪end=f"\n{'-'*100}\n")

    print("Final Validation Results")
    print(layer_tests_2[f"Test-{i}"]["training_results"].iloc[-1, 8:12],␣
↪end=f"\n{'-'*100}\n")

    print("Test Set Results")
    print(layer_tests_2[f"Test-{i}"]["error_metrics"].iloc[0],␣
↪end=f"\n{'='*100}\n\n\n\n")

    ax = layer_tests_2[f"Test-{i}"]["training_results"].plot(
        y=["rmse", "val_rmse"], title=f"Model Trained with Hidden Layer␣
↪Configuration (10, {i})",
    )
    ax.set_xlabel("Epochs")
    ax.set_ylabel("Root Mean Squared Error (RMSE)")
```

```
Model Trained with Hidden Layer Configuration (10, 1)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae         9.292601
mse       417.818817
r_sqr       0.862269
rmse       20.440617
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
```

```
val_mae        9.735302
val_mse      376.200705
val_r_sqr      0.878966
val_rmse      19.395894
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         407.018132
rmse         20.174690
mae          13.710399
r_sqr         0.875703
st_mse        0.003585
st_rmse       0.059873
st_mae        0.026779
st_r_sqr      0.810105
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 2)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        6.076114
mse      140.742164
r_sqr      0.959537
rmse      11.863480
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        6.630507
val_mse      157.705785
val_r_sqr      0.938995
val_rmse      12.558096
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         2379.356258
rmse          48.778646
mae           33.361029
r_sqr          0.039664
st_mse         0.001525
st_rmse        0.039051
```

```
st_mae         0.021237
st_r_sqr       0.937248
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 3)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.889117
mse      235.792069
r_sqr      0.926761
rmse      15.355522
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.498321
val_mse      221.075697
val_r_sqr      0.924965
val_rmse      14.868614
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         562.700580
rmse         23.721311
mae          15.513718
r_sqr         0.805169
st_mse        0.002758
st_rmse       0.052520
st_mae        0.026391
st_r_sqr      0.874321
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 4)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        6.909384
mse      157.518853
```

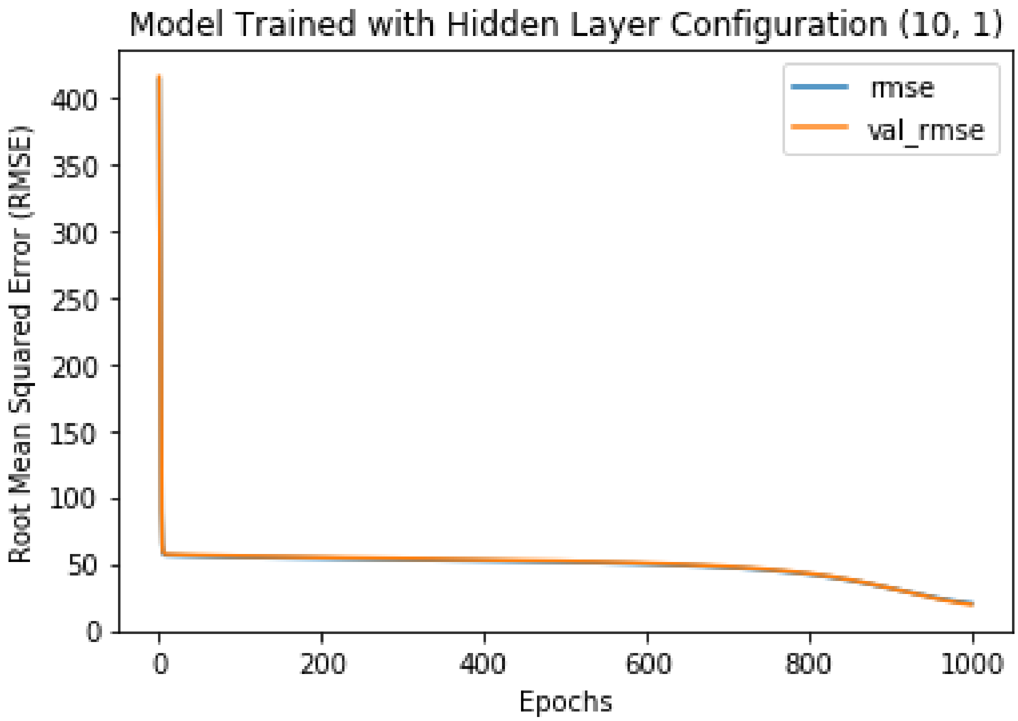
```
r_sqr      0.946891
rmse      12.550651
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        6.185888
val_mse      113.961663
val_r_sqr      0.957834
val_rmse      10.675283
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         219.828390
rmse         14.826611
mae           8.826869
r_sqr         0.943506
st_mse        0.001560
st_rmse       0.039499
st_mae        0.017900
st_r_sqr      0.919833
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 5)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae       10.522868
mse      311.979397
r_sqr      0.907433
rmse      17.662939
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.899300
val_mse      245.319428
val_r_sqr      0.902181
val_rmse      15.662676
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
```

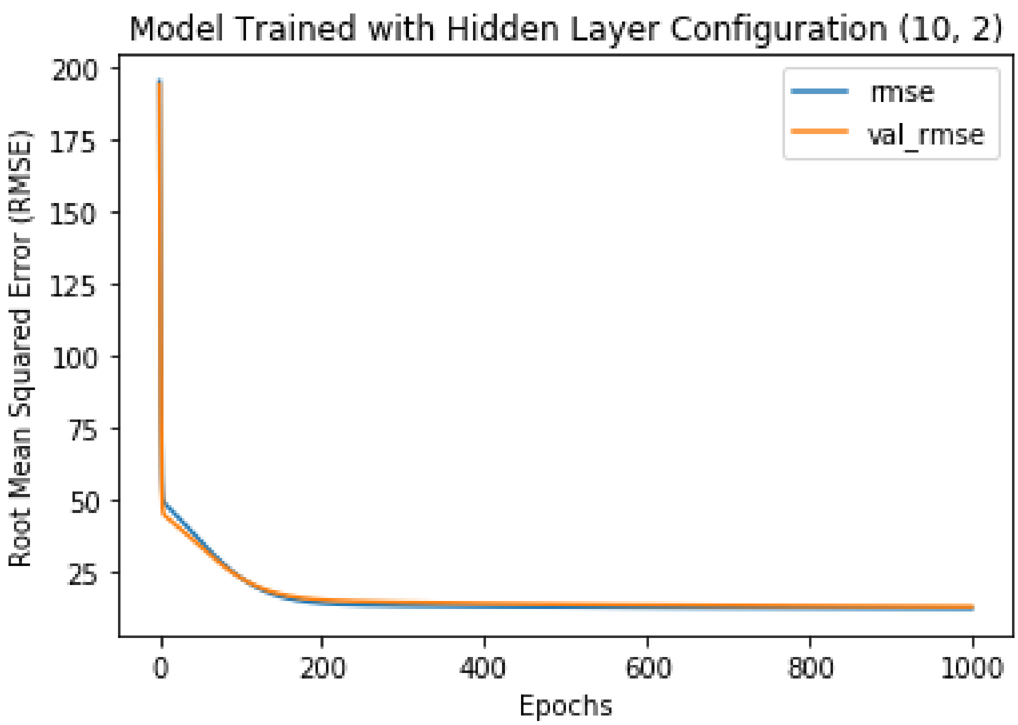
```
mse         537.153056
rmse         23.176563
mae          15.734755
r_sqr         0.812915
st_mse        0.001906
st_rmse       0.043662
st_mae        0.023878
st_r_sqr      0.866824
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 6)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        5.748496
mse      113.480440
r_sqr      0.958104
rmse      10.652720
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.020809
val_mse      193.100435
val_r_sqr      0.944236
val_rmse      13.896058
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         376.260625
rmse         19.397439
mae           9.536395
r_sqr         0.902956
st_mse        0.000709
st_rmse       0.026622
st_mae        0.014719
st_r_sqr      0.943632
Name: 0, dtype: float64
================================================================================
====================
```

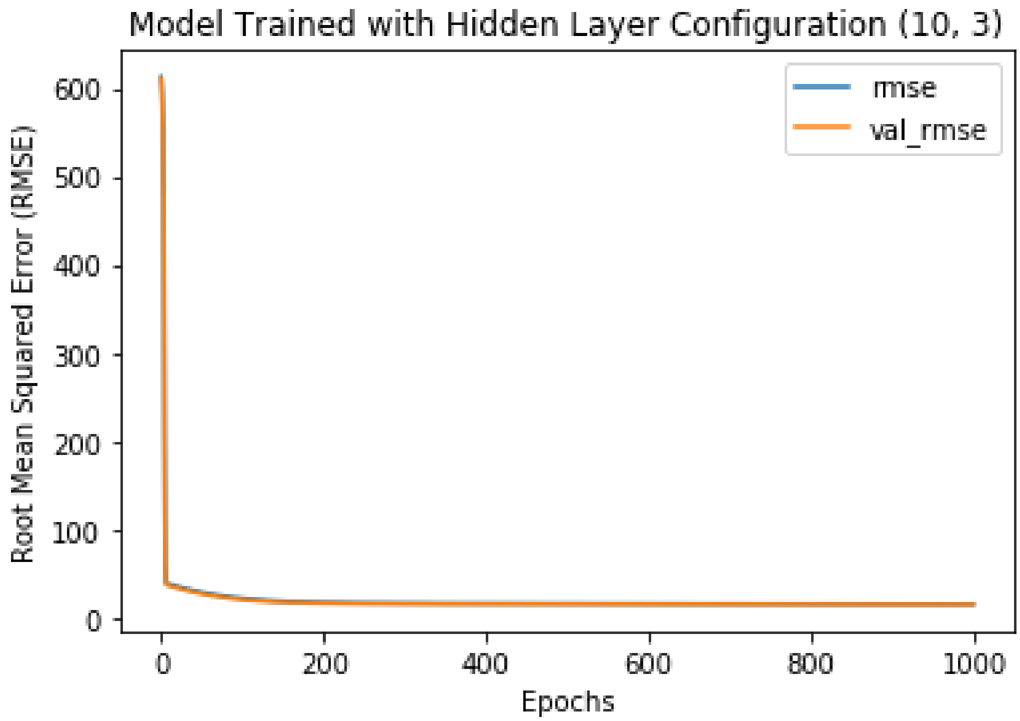
```
Model Trained with Hidden Layer Configuration (10, 7)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        6.601837
mse      131.016762
r_sqr      0.957874
rmse      11.446255
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.153569
val_mse      138.887926
val_r_sqr      0.946894
val_rmse      11.785072
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         375.751609
rmse         19.384313
mae          12.805029
r_sqr         0.893769
st_mse        0.001086
st_rmse       0.032958
st_mae        0.018602
st_r_sqr      0.938426
Name: 0, dtype: float64
================================================================================
====================


Model Trained with Hidden Layer Configuration (10, 8)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.958540
mse      197.719732
r_sqr      0.925305
rmse      14.061285
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.296390
val_mse      239.692717
```

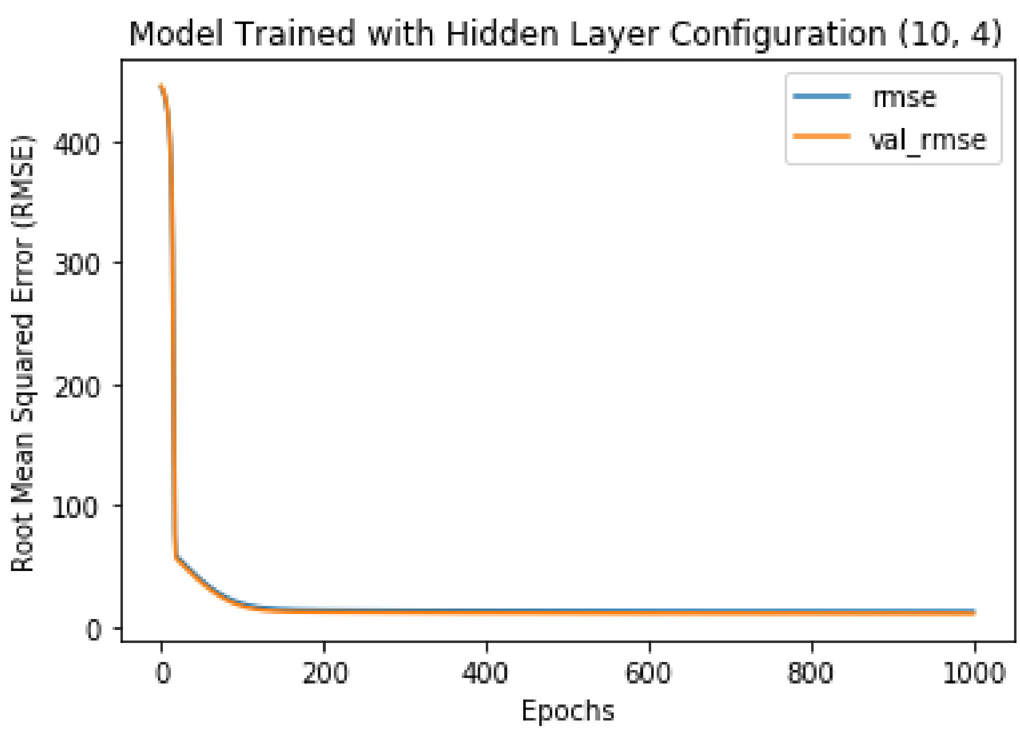
```
val_r_sqr      0.920791
val_rmse      15.482013
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         371.719131
rmse         19.280019
mae          11.847606
r_sqr         0.917240
st_mse        0.001597
st_rmse       0.039967
st_mae        0.020562
st_r_sqr      0.928691
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 9)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        6.954281
mse      151.877066
r_sqr      0.949522
rmse      12.323841
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.531540
val_mse      145.484447
val_r_sqr      0.957469
val_rmse      12.061693
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1538.547901
rmse          39.224328
mae           27.420260
r_sqr          0.494127
st_mse         0.001627
st_rmse        0.040332
st_mae         0.024097
st_r_sqr       0.922437
```

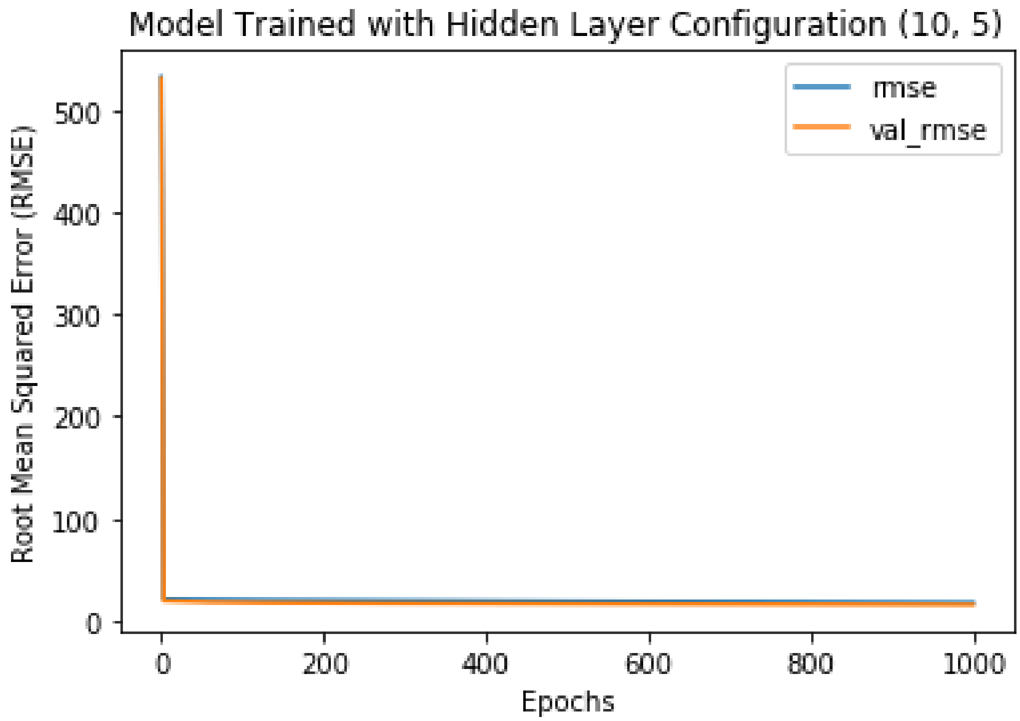
```
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        9.935704
mse      316.371832
r_sqr      0.902427
rmse      17.786844
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.591323
val_mse      259.981100
val_r_sqr      0.921527
val_rmse      16.123929
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         348.213726
rmse         18.660486
mae          11.980118
r_sqr         0.857762
st_mse        0.008356
st_rmse       0.091413
st_mae        0.047380
st_r_sqr      0.644567
Name: 0, dtype: float64
================================================================================
====================
```

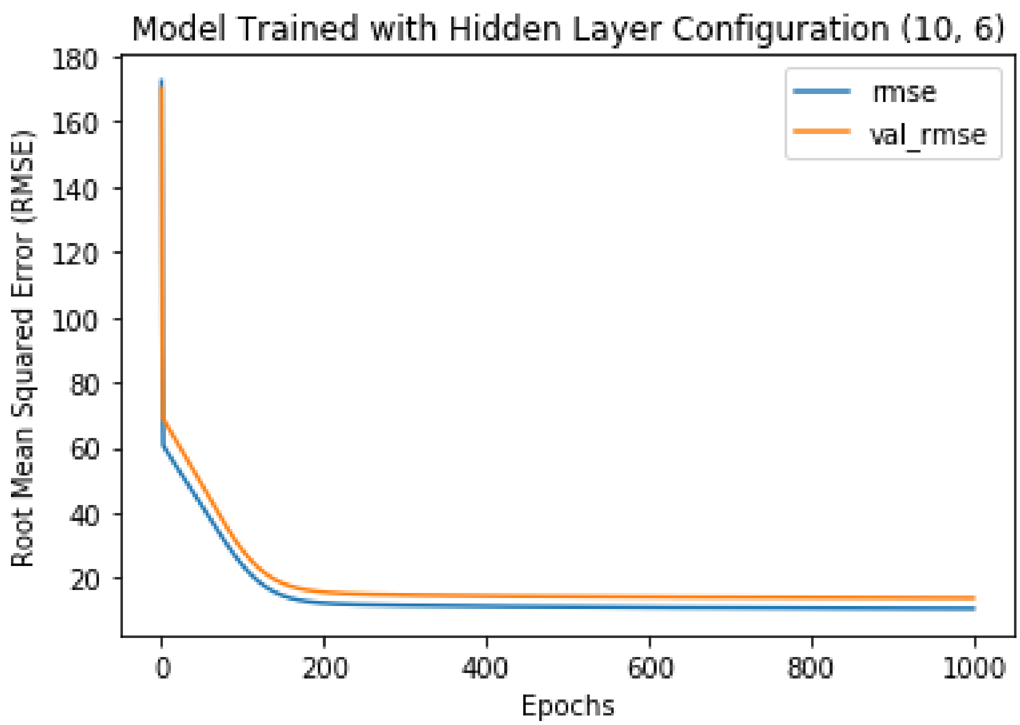

Model Trained with Hidden Layer Configuration (10, 1)
rmse
val_rmse
Root Mean Squared Error (RMSE)
400
350
300
250
200
150
100
50
0
0
200
400
600
800
1000
Epochs

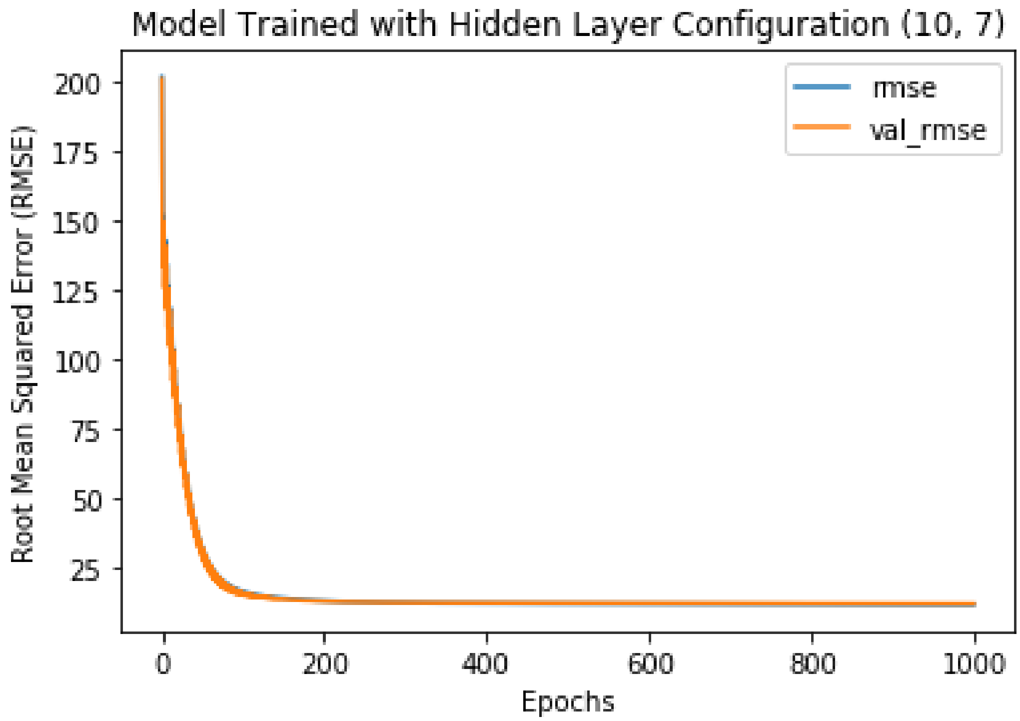

Model Trained with Hidden Layer Configuration (10, 2)
rmse
val_rmse
Root Mean Squared Error (RMSE)
200
175
150
125
100
75
50
25
0
200
400
600
800
1000
Epochs

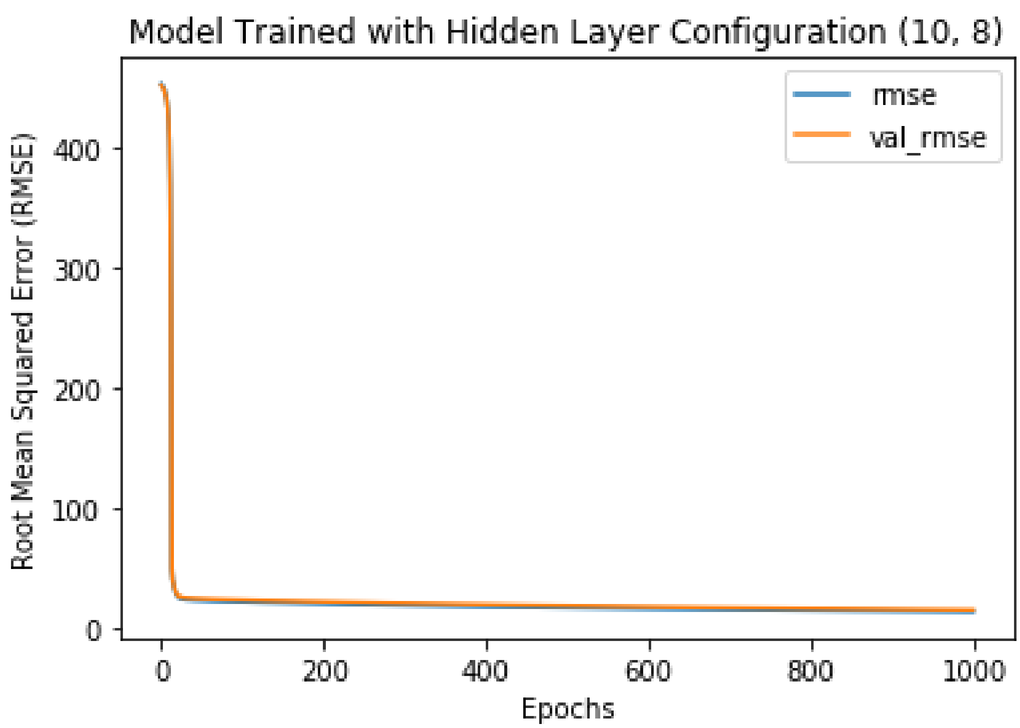

Model Trained with Hidden Layer Configuration (10, 3)
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
600
0
200
400
600
800
1000
Epochs

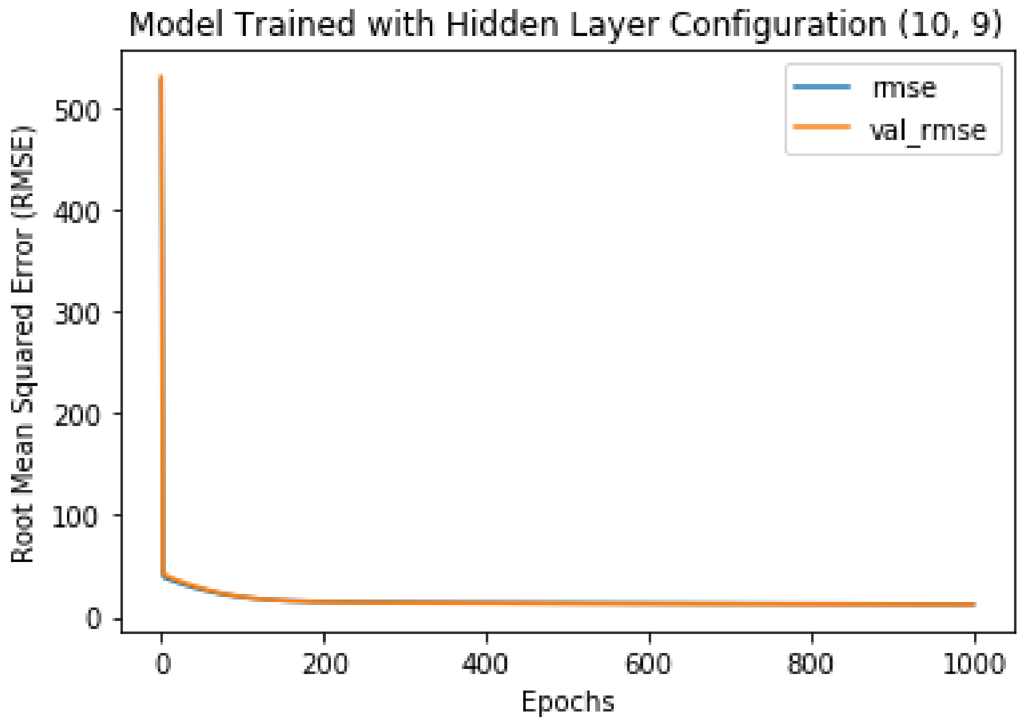

Model Trained with Hidden Layer Configuration (10, 4)
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
0
200
400
600
800
1000
Epochs

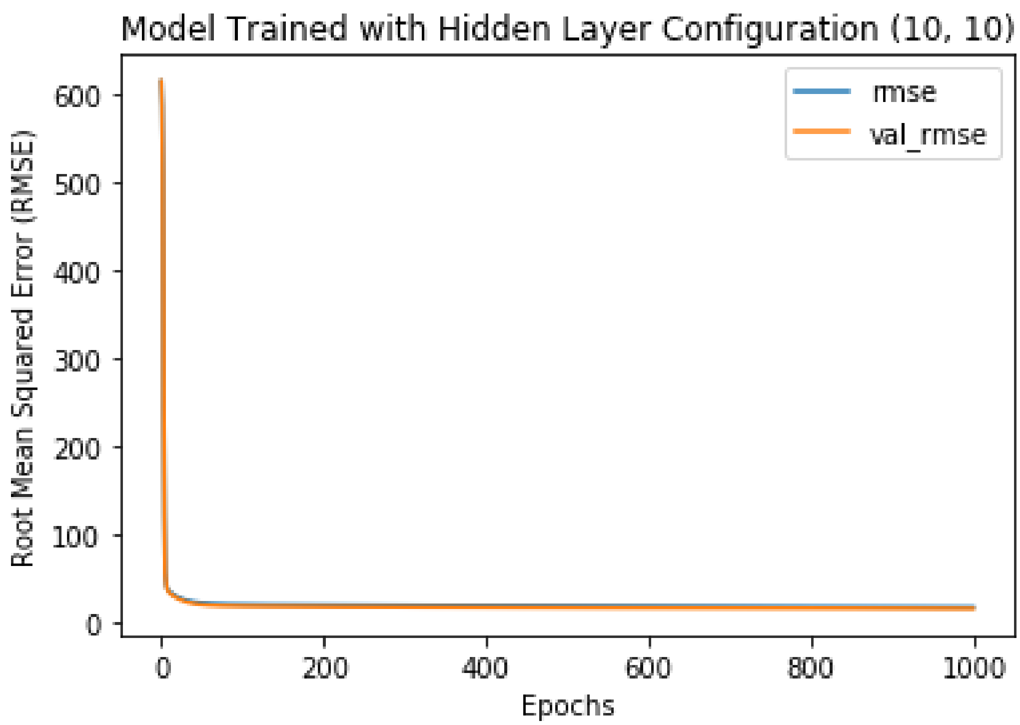

Model Trained with Hidden Layer Configuration (10, 5)
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (10, 6)
rmse
val_rmse
Root Mean Squared Error (RMSE)
20
40
60
80
100
120
140
160
180
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (10, 7)
rmse
val_rmse
Root Mean Squared Error (RMSE)
200
175
150
125
100
75
50
25
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (10, 8)
rmse
val_rmse
Root Mean Squared Error (RMSE)
400
300
200
100
0
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (10, 9)
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (10, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
600
0
200
400
600
800
1000
Epochs

[19]:
```python
layer_tests_3 = dict()
for i in range(1, 11):
    layer_tests_3[f"Test-{i}"] = build_train_test(
        fs,
        feature_cols,
        target_cols,
        layers=("auto", i, 10, 1),
        activ_func="tanh",
        epochs=1000,
        l_rate=0.3
    )
```

[20]:
```python
for i in range(1,11):
    print(f"Model Trained with Hidden Layer Configuration ({i}, 10)",␣
↪end=f"\n{'-'*100}\n")

    print("Final Training Results")
    print(layer_tests_3[f"Test-{i}"]["training_results"].iloc[-1, :4],␣
↪end=f"\n{'-'*100}\n")

    print("Final Validation Results")
    print(layer_tests_3[f"Test-{i}"]["training_results"].iloc[-1, 8:12],␣
↪end=f"\n{'-'*100}\n")

    print("Test Set Results")
    print(layer_tests_3[f"Test-{i}"]["error_metrics"].iloc[0],␣
↪end=f"\n{'='*100}\n\n\n\n")

    ax = layer_tests_3[f"Test-{i}"]["training_results"].plot(
        y=["rmse", "val_rmse"], title=f"Model Trained with Hidden Layer␣
↪Configuration ({i}, 10)",
    )
    ax.set_xlabel("Epochs")
    ax.set_ylabel("Root Mean Squared Error (RMSE)")
```

```
Model Trained with Hidden Layer Configuration (1, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae         9.687211
mse       322.933810
r_sqr       0.891352
rmse       17.970359
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
```

```
val_mae        9.022622
val_mse      262.254092
val_r_sqr      0.907723
val_rmse      16.194261
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         415.454993
rmse         20.382713
mae          11.214236
r_sqr         0.887585
st_mse        0.003672
st_rmse       0.060598
st_mae        0.032000
st_r_sqr      0.800421
Name: 0, dtype: float64
================================================================================
====================


Model Trained with Hidden Layer Configuration (2, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.003304
mse      160.427402
r_sqr      0.942346
rmse      12.665994
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.080296
val_mse      250.315750
val_r_sqr      0.935941
val_rmse      15.821370
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         306.903190
rmse         17.518653
mae          11.458790
r_sqr         0.902450
st_mse        0.003250
st_rmse       0.057008
```

```
st_mae        0.028263
st_r_sqr      0.864117
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (3, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        9.188204
mse      257.334761
r_sqr      0.914666
rmse      16.041657
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.752097
val_mse      177.811155
val_r_sqr      0.918427
val_rmse      13.334585
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         561.650211
rmse         23.699161
mae          16.234630
r_sqr         0.866639
st_mse        0.004243
st_rmse       0.065135
st_mae        0.029878
st_r_sqr      0.843230
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (4, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.394178
mse      171.687990
```

```
r_sqr      0.945897
rmse      13.102976
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.515336
val_mse      180.175509
val_r_sqr      0.937592
val_rmse      13.422947
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         333.976598
rmse         18.275027
mae          11.292514
r_sqr         0.891780
st_mse        0.002213
st_rmse       0.047044
st_mae        0.022497
st_r_sqr      0.889996
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (5, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.882073
mse      193.996556
r_sqr      0.937152
rmse      13.928265
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.610485
val_mse      185.159962
val_r_sqr      0.945962
val_rmse      13.607350
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
```

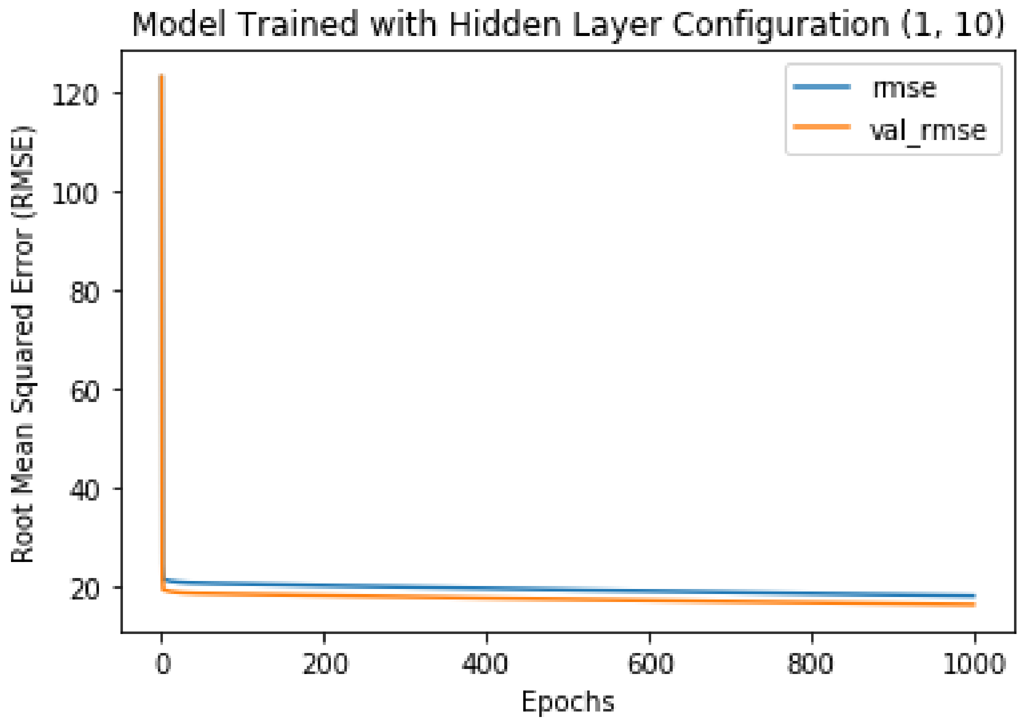
```
mse         1107.365880
rmse          33.277107
mae           22.350471
r_sqr          0.606026
st_mse         0.001876
st_rmse        0.043311
st_mae         0.024061
st_r_sqr       0.895976
Name: 0, dtype: float64
================================================================================
====================
```

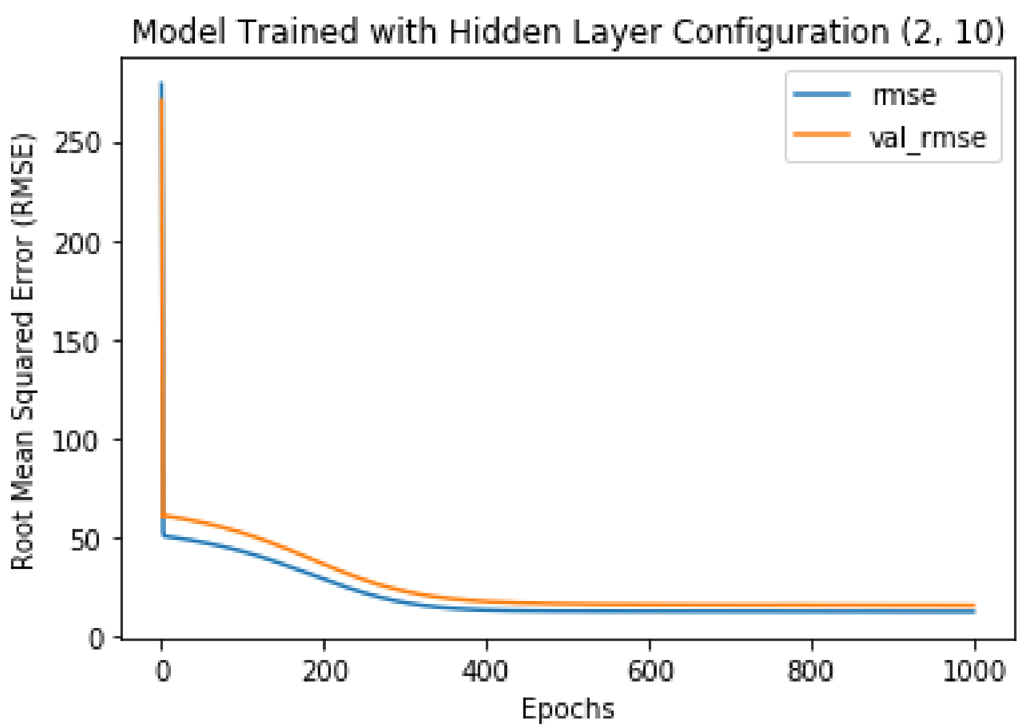
```
Model Trained with Hidden Layer Configuration (6, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.222177
mse      202.121727
r_sqr      0.926889
rmse      14.216952
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.057300
val_mse      165.134807
val_r_sqr      0.944909
val_rmse      12.850479
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         237.158358
rmse         15.399947
mae           9.838986
r_sqr         0.942777
st_mse        0.002350
st_rmse       0.048479
st_mae        0.025716
st_r_sqr      0.886089
Name: 0, dtype: float64
================================================================================
====================
```

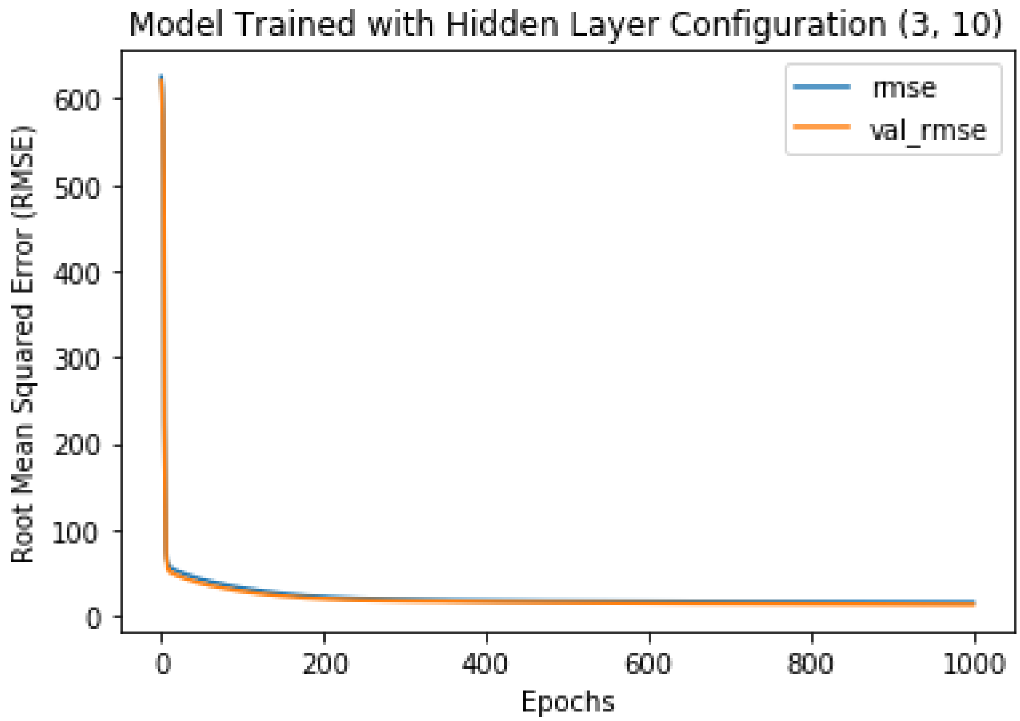
```
Model Trained with Hidden Layer Configuration (7, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.145765
mse      150.924877
r_sqr      0.953200
rmse      12.285149
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        5.852051
val_mse      106.614364
val_r_sqr      0.962687
val_rmse      10.325423
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         226.863857
rmse         15.062000
mae           7.475933
r_sqr         0.923382
st_mse        0.000753
st_rmse       0.027448
st_mae        0.016128
st_r_sqr      0.921576
Name: 0, dtype: float64
================================================================================
====================


Model Trained with Hidden Layer Configuration (8, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        5.884373
mse      127.769140
r_sqr      0.958400
rmse      11.303501
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        6.984307
val_mse      143.126778
```

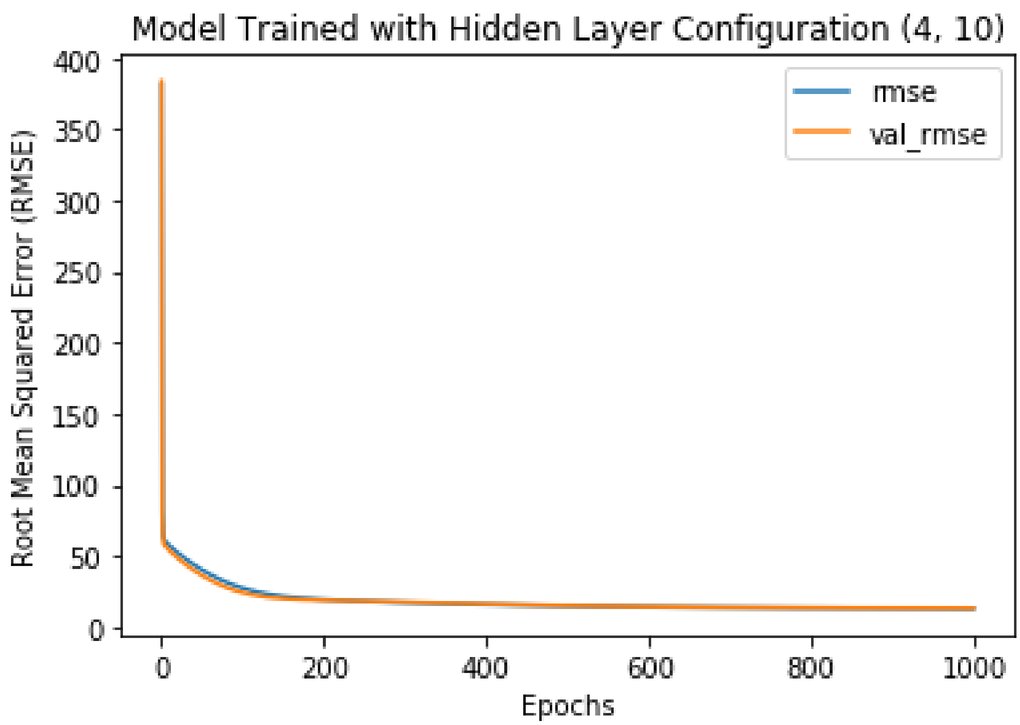
```
val_r_sqr      0.956033
val_rmse      11.963560
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1251.604582
rmse          35.378024
mae           24.385259
r_sqr          0.581258
st_mse         0.001336
st_rmse        0.036557
st_mae         0.019537
st_r_sqr       0.934907
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (9, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.540181
mse      213.223943
r_sqr      0.928560
rmse      14.602190
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae       10.470772
val_mse      331.918736
val_r_sqr      0.887989
val_rmse      18.218637
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         484.287486
rmse         22.006533
mae          11.519720
r_sqr         0.864286
st_mse        0.000800
st_rmse       0.028284
st_mae        0.017758
st_r_sqr      0.930837
```

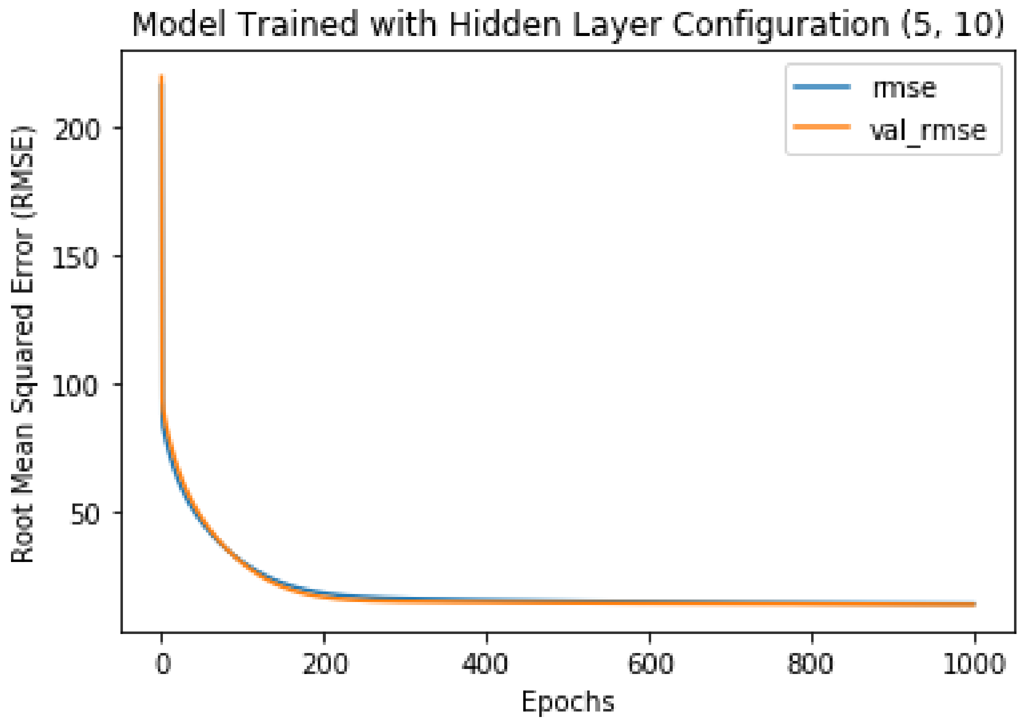
```
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        6.367745
mse      139.122132
r_sqr      0.956971
rmse      11.795005
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        6.180989
val_mse      119.222941
val_r_sqr      0.960460
val_rmse      10.918926
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1062.005232
rmse          32.588422
mae           21.931587
r_sqr          0.617991
st_mse         0.001747
st_rmse        0.041801
st_mae         0.020720
st_r_sqr       0.926734
Name: 0, dtype: float64
================================================================================
====================
```

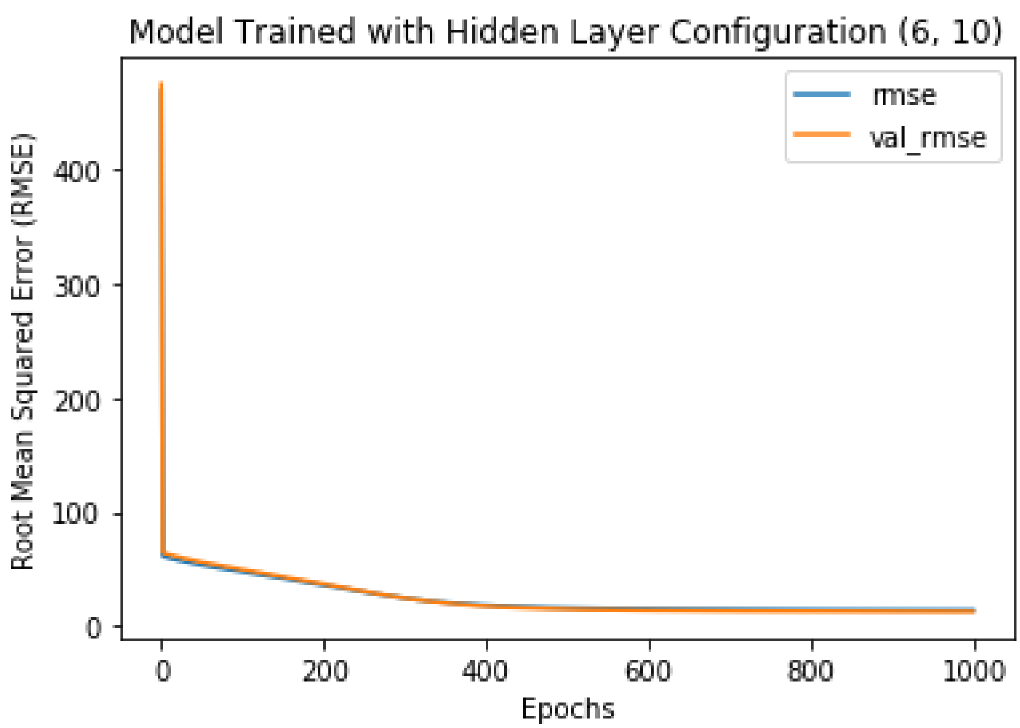

Model Trained with Hidden Layer Configuration (1, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
120
100
80
60
40
20
0
200
400
600
800
1000

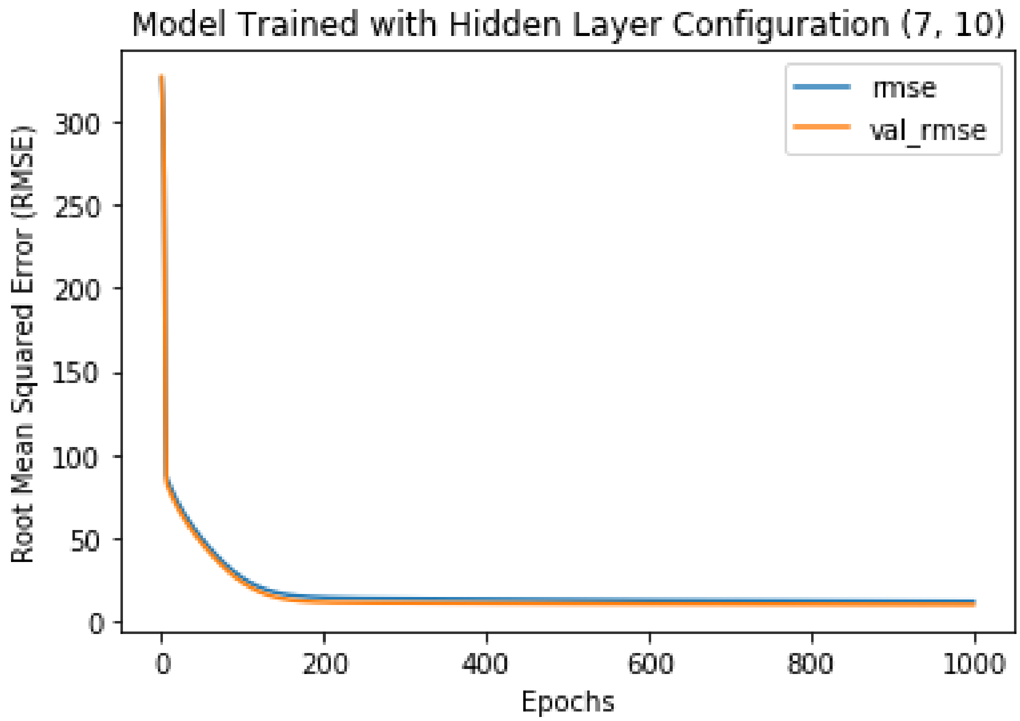

Model Trained with Hidden Layer Configuration (2, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
250
200
150
100
50
0
0
200
400
600
800
1000

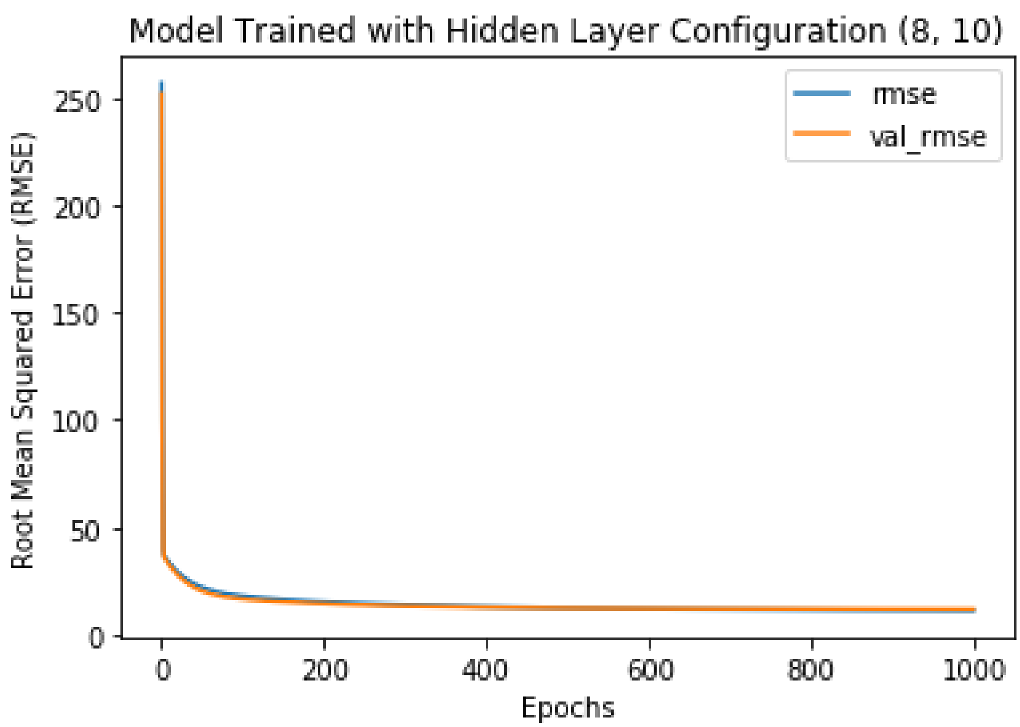

Model Trained with Hidden Layer Configuration (3, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
600
0
200
400
600
800
1000
Epochs

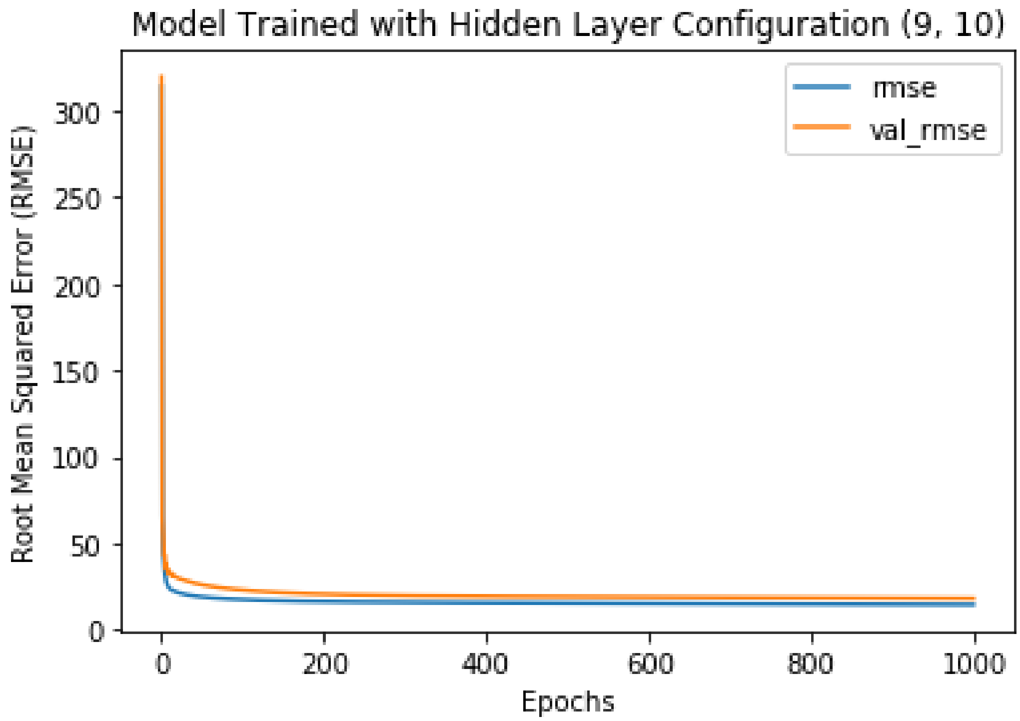

Model Trained with Hidden Layer Configuration (4, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
50
100
150
200
250
300
350
400
0
200
400
600
800
1000
Epochs

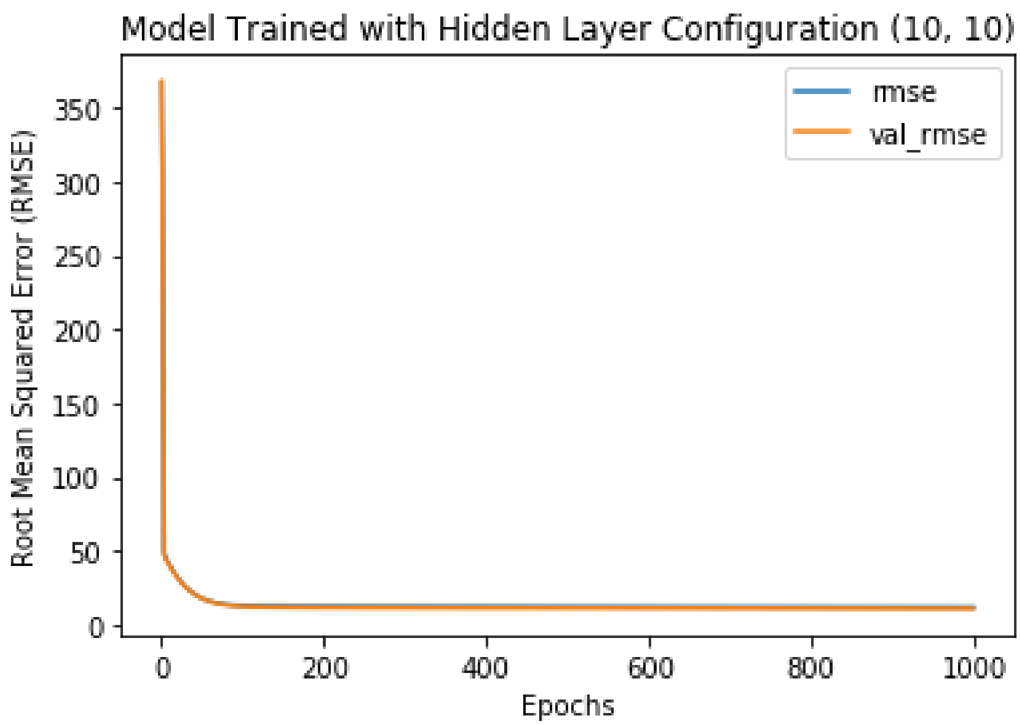

Model Trained with Hidden Layer Configuration (5, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
200
150
100
50
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (6, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
400
300
200
100
0
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (7, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
50
100
150
200
250
300
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (8, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
50
100
150
200
250
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (9, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
300
250
200
150
100
50
0
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (10, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
350
300
250
200
150
100
50
0
0
200
400
600
800
1000
Epochs

```
[ ]:
```