Appendix 4
Implementation of MLP Algorithm and Development and Training of MLP Model

[1]:
```
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]:
```
# 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]: # 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]: ## 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),
```

```python
                "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

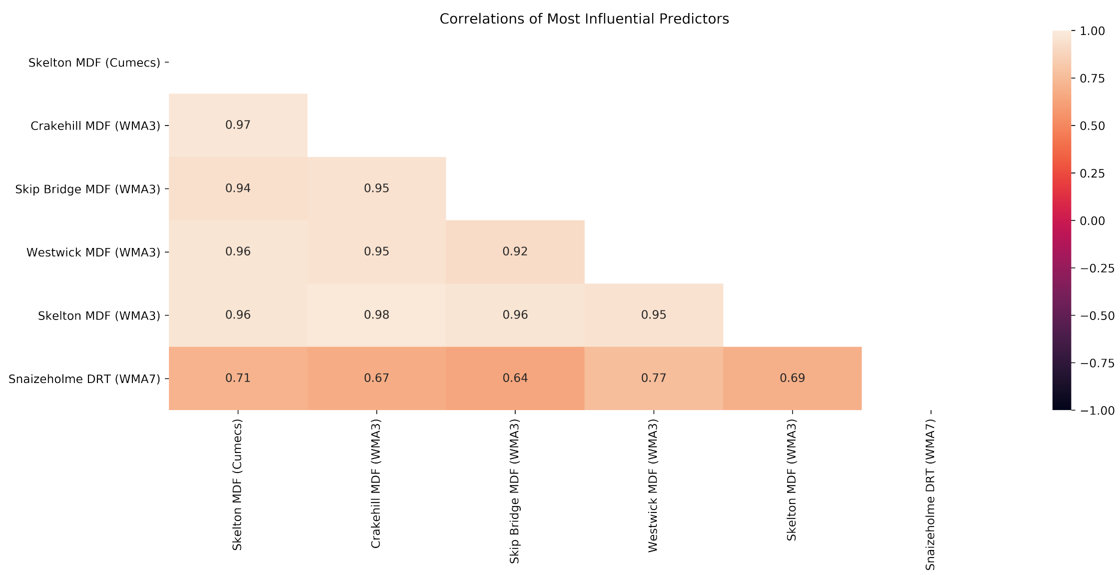
Building Feature Sets

[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       11.115425
mse      298.617903
r_sqr      0.907014
rmse      17.280564
```

```
Name: 499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae       11.824059
val_mse      355.111826
val_r_sqr      0.899109
val_rmse      18.844411
Name: 499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         540.297189
rmse         23.244294
mae          18.232164
r_sqr         0.766672
st_mse        0.001970
st_rmse       0.044380
st_mae        0.025539
st_r_sqr      0.888120
Name: 0, dtype: float64
================================================================================
====================



Model trained with 1000 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        8.688907
mse      212.143011
r_sqr      0.937055
rmse      14.565130
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.612176
val_mse      224.111494
val_r_sqr      0.928040
val_rmse      14.970354
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         263.936692
rmse         16.246129
```

```
mae           9.433544
r_sqr         0.880873
st_mse        0.000677
st_rmse       0.026011
st_mae        0.016009
st_r_sqr      0.938698
Name: 0, dtype: float64
================================================================================
====================



Model trained with 1500 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        7.725300
mse      180.929892
r_sqr      0.939766
rmse      13.451018
Name: 1499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.400230
val_mse      230.727938
val_r_sqr      0.930766
val_rmse      15.189731
Name: 1499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         763.629961
rmse         27.633855
mae          17.428569
r_sqr         0.756392
st_mse        0.001362
st_rmse       0.036908
st_mae        0.019721
st_r_sqr      0.933314
Name: 0, dtype: float64
================================================================================
====================



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

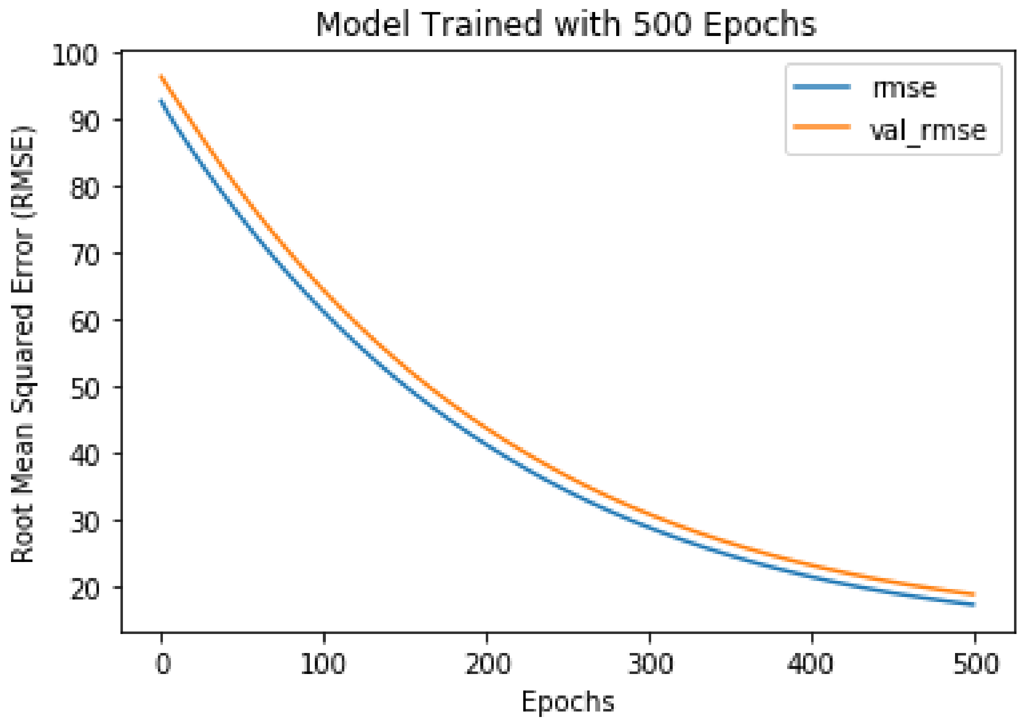
```
--------------------
Final Training results
mae        9.859058
mse      323.622713
r_sqr      0.903529
rmse      17.989517
Name: 1999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        9.298527
val_mse      262.621712
val_r_sqr      0.897150
val_rmse      16.205607
Name: 1999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1534.300343
rmse          39.170146
mae           23.837348
r_sqr          0.465289
st_mse         0.001491
st_rmse        0.038611
st_mae         0.023668
st_r_sqr       0.919180
Name: 0, dtype: float64
================================================================================
====================



Model trained with 2500 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae       10.612285
mse      311.711768
r_sqr      0.905703
rmse      17.655361
Name: 2499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        9.807566
val_mse      261.335346
val_r_sqr      0.908631
val_rmse      16.165870
```

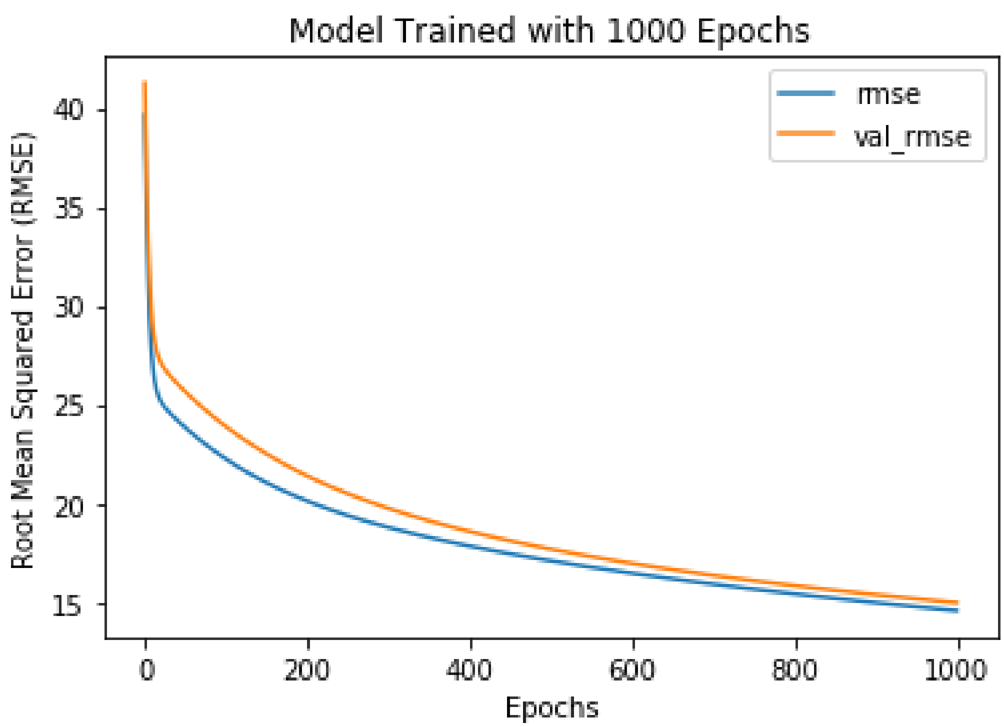
```
Name: 2499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         2619.121607
rmse          51.177354
mae           32.500738
r_sqr          0.035265
st_mse         0.003658
st_rmse        0.060480
st_mae         0.037223
st_r_sqr       0.766383
Name: 0, dtype: float64
================================================================================
====================



Model trained with 3000 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.129363
mse      124.789594
r_sqr      0.960619
rmse      11.170926
Name: 2999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        6.326957
val_mse      124.882068
val_r_sqr      0.952840
val_rmse      11.175065
Name: 2999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         648.318969
rmse         25.462109
mae          14.113753
r_sqr         0.802339
st_mse        0.001672
st_rmse       0.040895
st_mae        0.024533
st_r_sqr      0.926110
Name: 0, dtype: float64
================================================================================
```

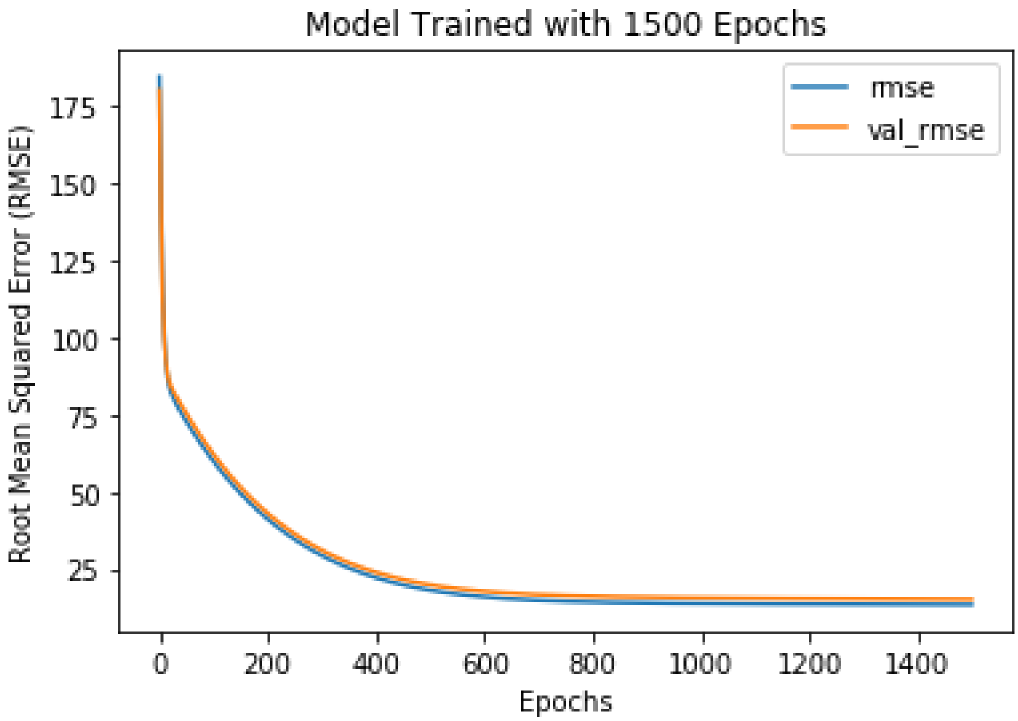
```
====================


Model trained with 3500 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        8.736078
mse      250.379080
r_sqr      0.920063
rmse      15.823371
Name: 3499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        9.033819
val_mse      256.454614
val_r_sqr      0.921544
val_rmse      16.014200
Name: 3499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         313.532026
rmse         17.706836
mae          10.328859
r_sqr         0.888670
st_mse        0.004751
st_rmse       0.068925
st_mae        0.038536
st_r_sqr      0.740614
Name: 0, dtype: float64
================================================================================
====================


Model trained with 4000 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.670029
mse      157.370618
r_sqr      0.949555
rmse      12.544745
Name: 3999, dtype: float64
--------------------------------------------------------------------------------
```

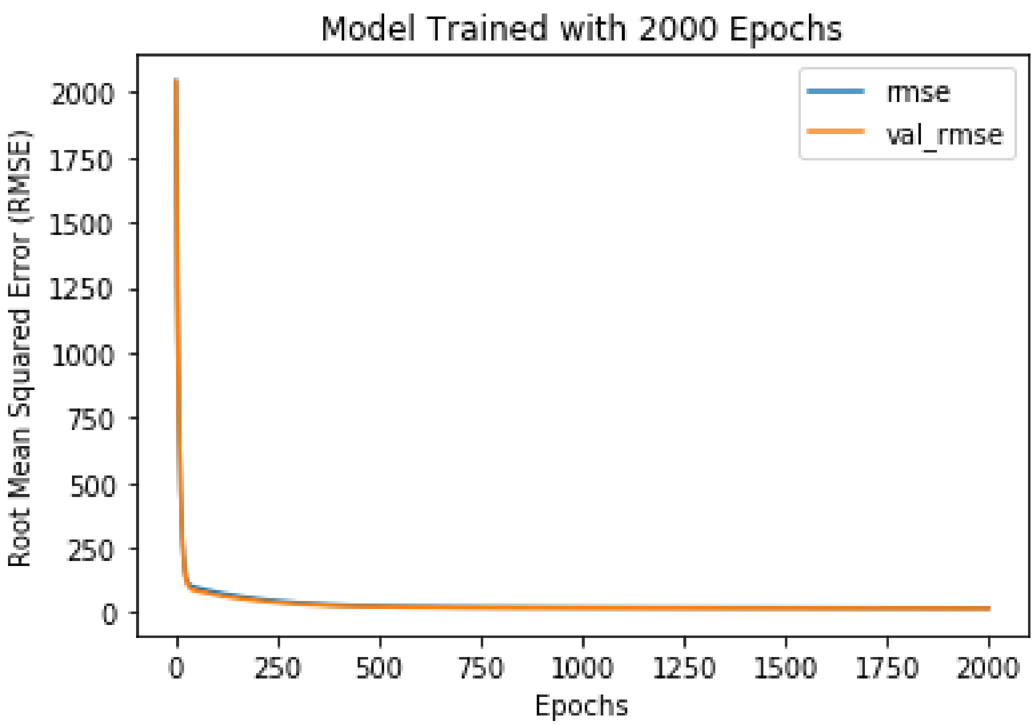
```
--------------------
Final Validation results
val_mae        6.234223
val_mse      178.456109
val_r_sqr      0.935996
val_rmse      13.358747
Name: 3999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         212.698633
rmse         14.584191
mae           7.794754
r_sqr         0.936354
st_mse        0.000564
st_rmse       0.023739
st_mae        0.013208
st_r_sqr      0.948000
Name: 0, dtype: float64
================================================================================
====================



Model trained with 4500 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.500001
mse      155.093720
r_sqr      0.946919
rmse      12.453663
Name: 4499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        6.386406
val_mse      131.872036
val_r_sqr      0.959946
val_rmse      11.483555
Name: 4499, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         932.030656
rmse         30.529177
mae          17.138363
r_sqr         0.725804
```

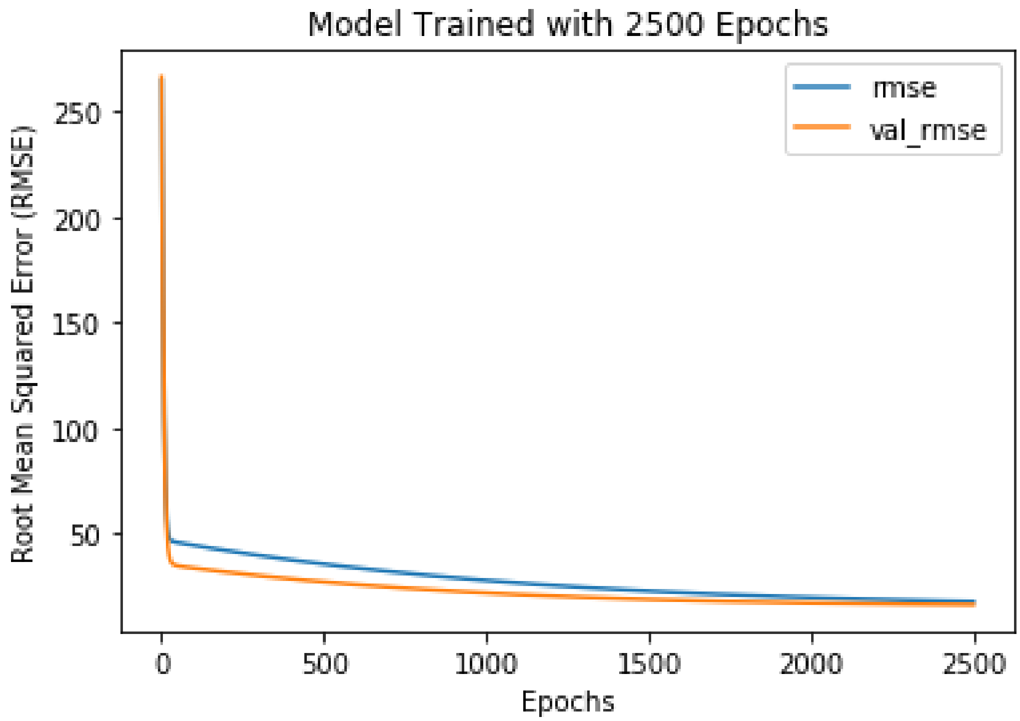
```
st_mse        0.001296
st_rmse       0.036001
st_mae        0.020815
st_r_sqr      0.940711
Name: 0, dtype: float64
================================================================================
====================



Model trained with 5000 epochs
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        8.351984
mse      199.404995
r_sqr      0.934940
rmse      14.121083
Name: 4999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.269012
val_mse      176.502604
val_r_sqr      0.949772
val_rmse      13.285428
Name: 4999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         830.270766
rmse         28.814419
mae          15.929263
r_sqr         0.702218
st_mse        0.005381
st_rmse       0.073353
st_mae        0.039558
st_r_sqr      0.794596
Name: 0, dtype: float64
================================================================================
====================
```

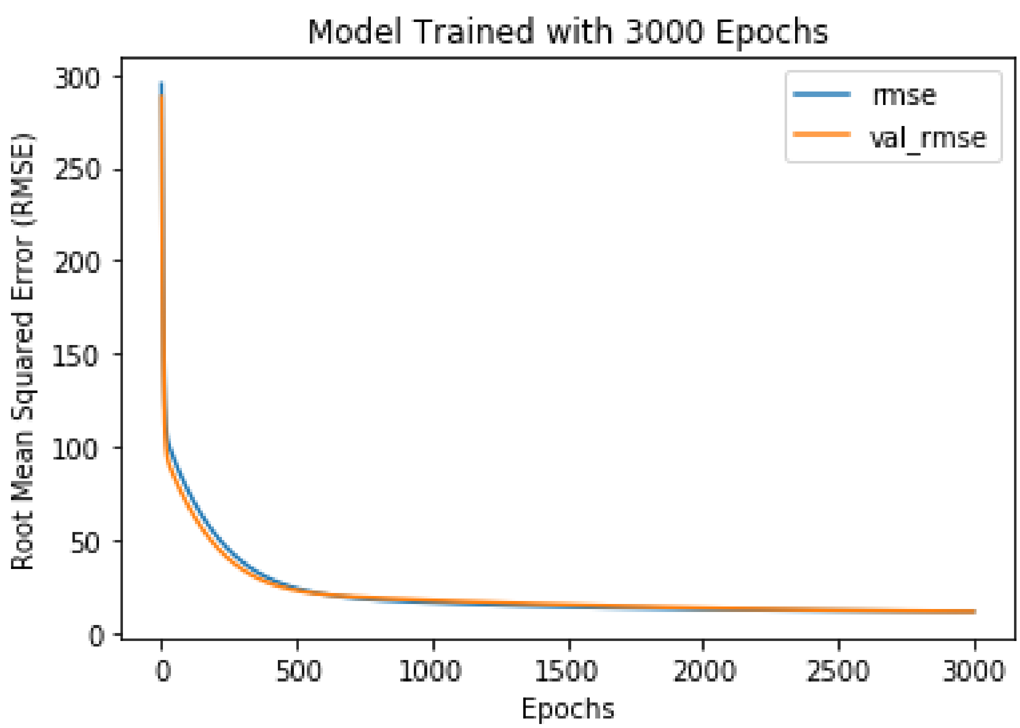

Model Trained with 500 Epochs
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
100
90
80
70
60
50
40
30
20
0
100
200
300
400
500

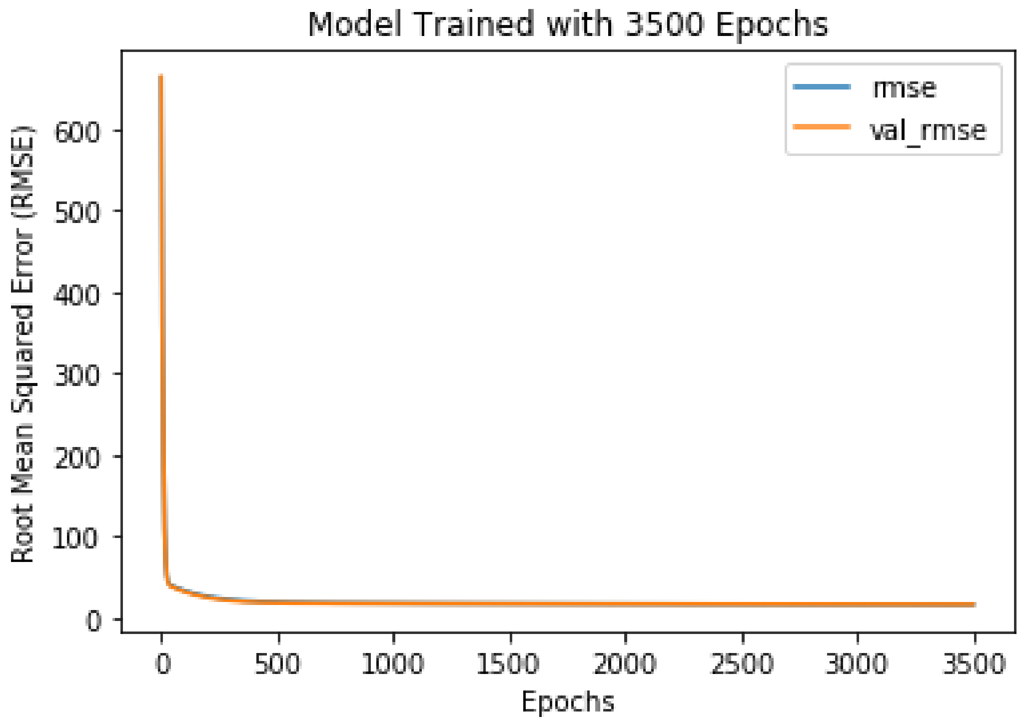

Model Trained with 1000 Epochs
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
40
35
30
25
20
15
0
200
400
600
800
1000

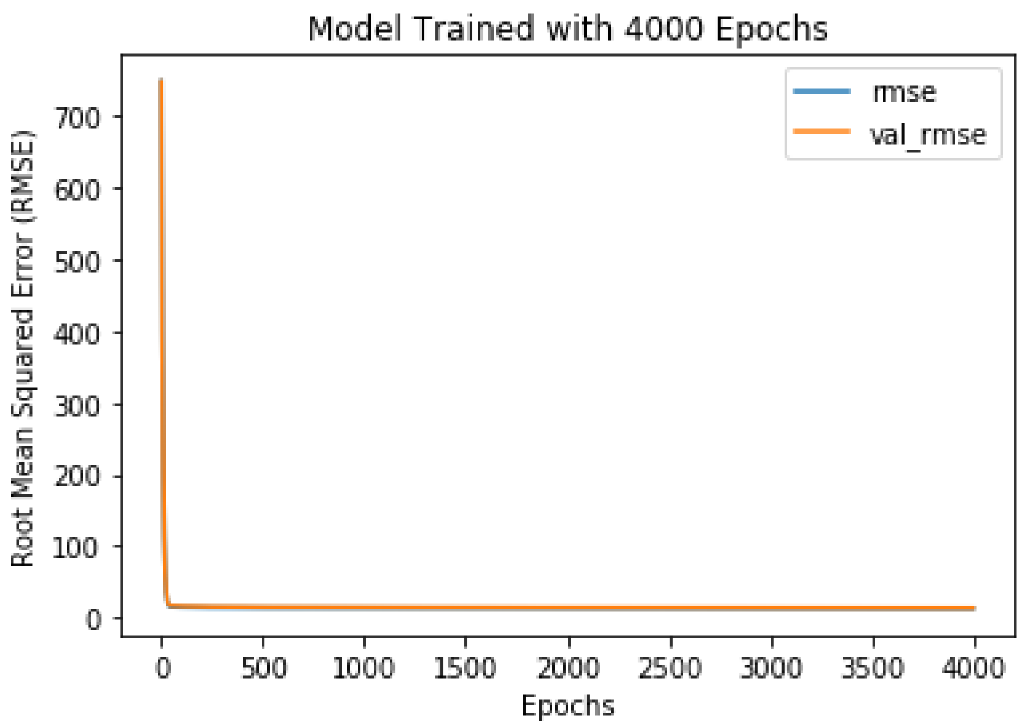

Model Trained with 1500 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
175
150
125
100
75
50
25
0
200
400
600
800
1000
1200
1400
Epochs

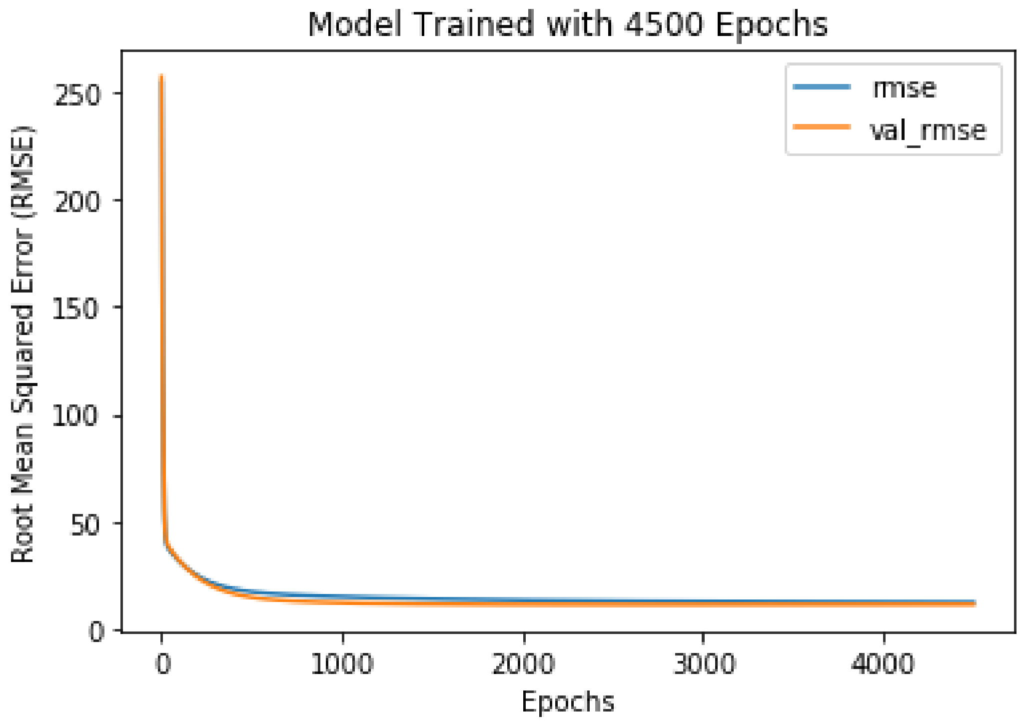

Model Trained with 2000 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
2000
1750
1500
1250
1000
750
500
250
0
0
250
500
750
1000
1250
1500
1750
2000
Epochs

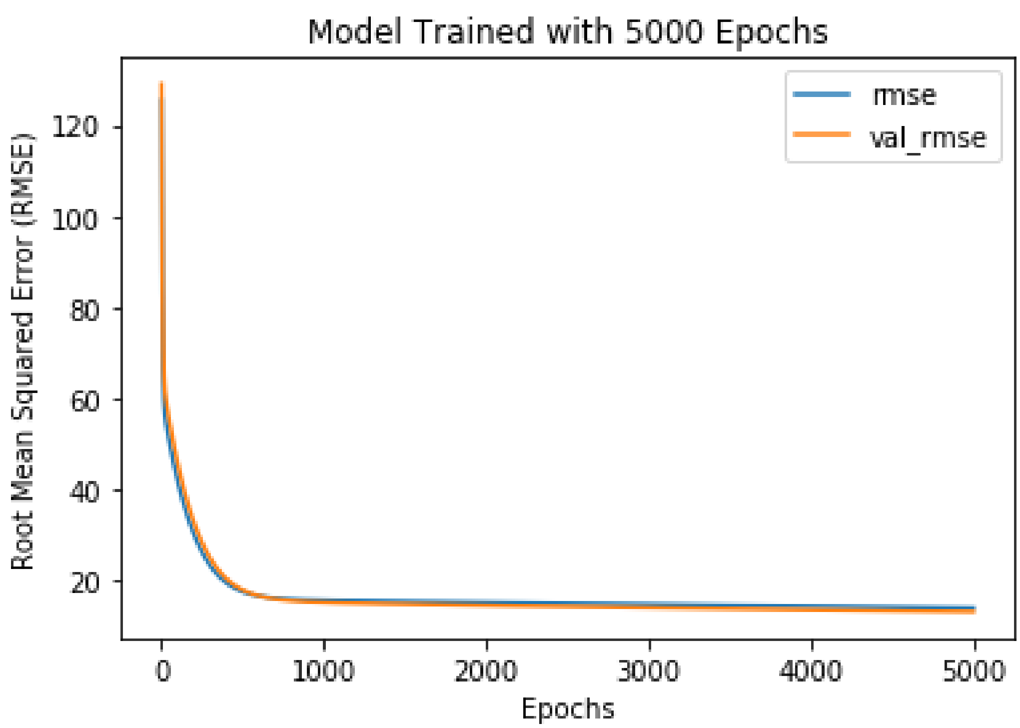

Model Trained with 2500 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
250
200
150
100
50
0
500
1000
1500
2000
2500
Epochs


Model Trained with 3000 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
300
250
200
150
100
50
0
0
500
1000
1500
2000
2500
3000
Epochs


Model Trained with 3500 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
600
0
500
1000
1500
2000
2500
3000
3500
Epochs


Model Trained with 4000 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
600
700
0
500
1000
1500
2000
2500
3000
3500
4000
Epochs


Model Trained with 4500 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
250
200
150
100
50
0
0
1000
2000
3000
4000
Epochs


Model Trained with 5000 Epochs
rmse
val_rmse
Root Mean Squared Error (RMSE)
120
100
80
60
40
20
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         6.728895
mse       144.544700
r_sqr       0.954585
rmse       12.022674
Name: 999, dtype: float64
--------------------------------------------------------------------------------
```

```
--------------------
Final Validation results
val_mae        6.827042
val_mse      154.903146
val_r_sqr      0.951899
val_rmse      12.446009
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         166.056780
rmse         12.886302
mae           6.526054
r_sqr         0.938928
st_mse        0.001043
st_rmse       0.032299
st_mae        0.015748
st_r_sqr      0.923022
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.2
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        9.262914
mse      288.251083
r_sqr      0.913986
rmse      16.977959
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.271171
val_mse      240.247581
val_r_sqr      0.909595
val_rmse      15.499922
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         361.894289
rmse         19.023519
mae           9.305752
r_sqr         0.867981
```

```
st_mse        0.004561
st_rmse       0.067532
st_mae        0.035913
st_r_sqr      0.800938
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.30000000000000004
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        8.066564
mse      204.581414
r_sqr      0.930061
rmse      14.303196
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.145104
val_mse      229.466800
val_r_sqr      0.916561
val_rmse      15.148162
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         236.679537
rmse         15.384393
mae           9.438774
r_sqr         0.939067
st_mse        0.001687
st_rmse       0.041069
st_mae        0.024278
st_r_sqr      0.912746
Name: 0, dtype: float64
================================================================================
====================



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

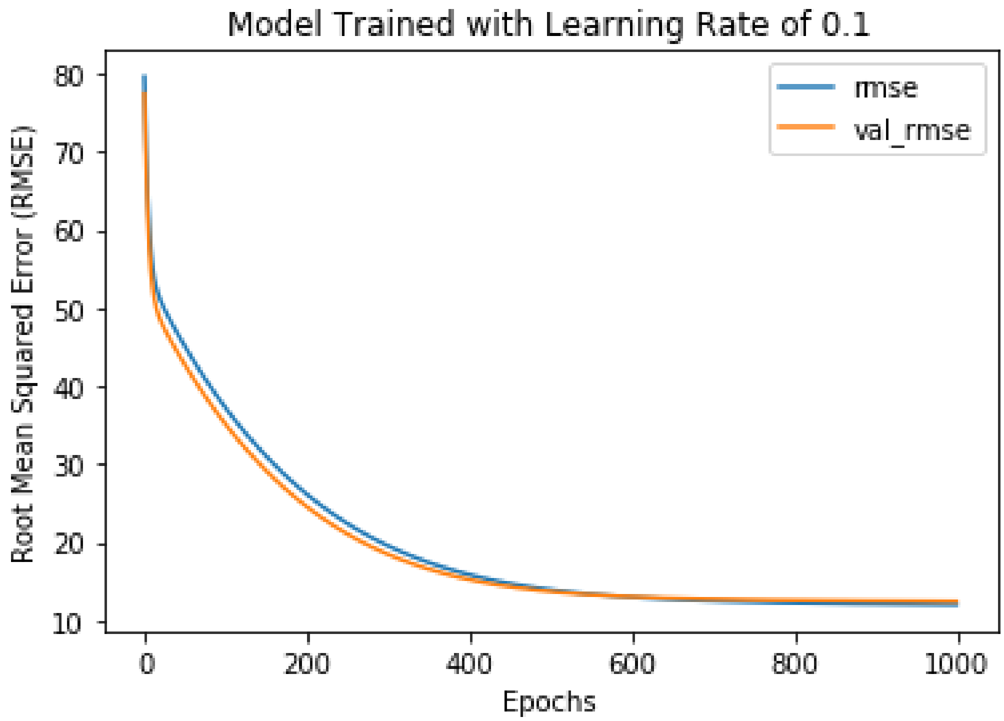
```
mae        8.032468
mse      223.892823
r_sqr      0.926797
rmse      14.963049
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        8.067513
val_mse      192.114710
val_r_sqr      0.934863
val_rmse      13.860545
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         359.787579
rmse         18.968067
mae          11.404283
r_sqr         0.892779
st_mse        0.001621
st_rmse       0.040257
st_mae        0.021636
st_r_sqr      0.916093
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.5
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.481272
mse      137.914126
r_sqr      0.955400
rmse      11.743685
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        6.547086
val_mse      172.664087
val_r_sqr      0.950973
val_rmse      13.140171
Name: 999, dtype: float64
--------------------------------------------------------------------------------
```

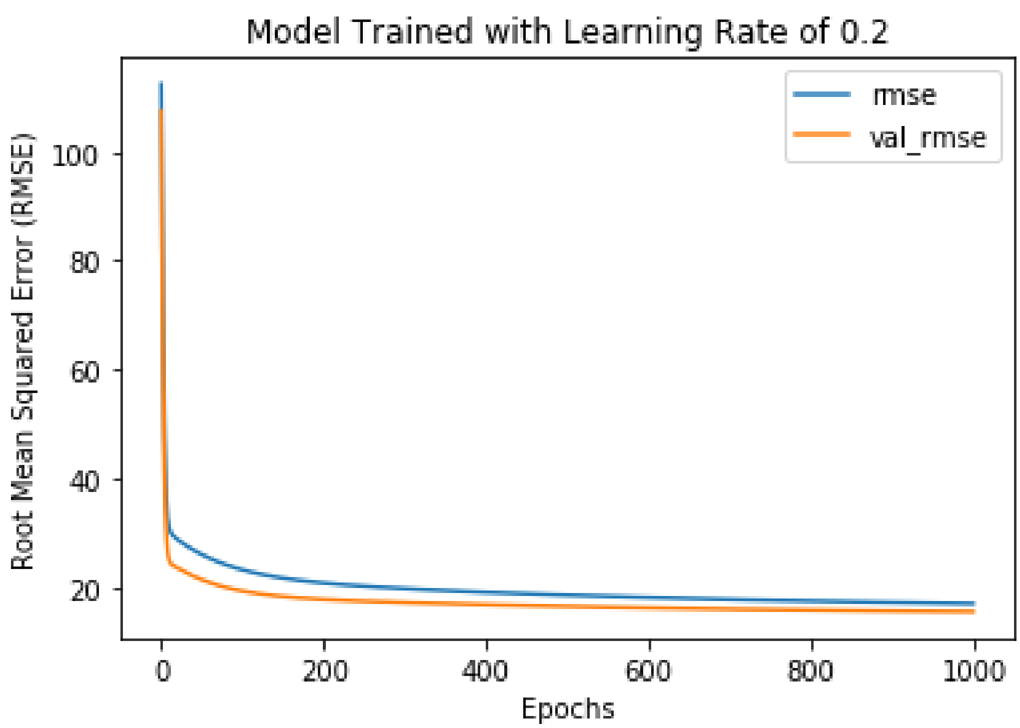
```
--------------------
Test Set Results
mse         1857.773818
rmse          43.101900
mae           26.614465
r_sqr          0.310309
st_mse         0.001140
st_rmse        0.033761
st_mae         0.019458
st_r_sqr       0.938681
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.6000000000000001
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.137523
mse      131.414319
r_sqr      0.959792
rmse      11.463608
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        5.821425
val_mse      132.483197
val_r_sqr      0.956994
val_rmse      11.510135
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1153.281994
rmse          33.960006
mae           20.625440
r_sqr          0.555230
st_mse         0.000767
st_rmse        0.027698
st_mae         0.015266
st_r_sqr       0.957287
Name: 0, dtype: float64
================================================================================
====================
```

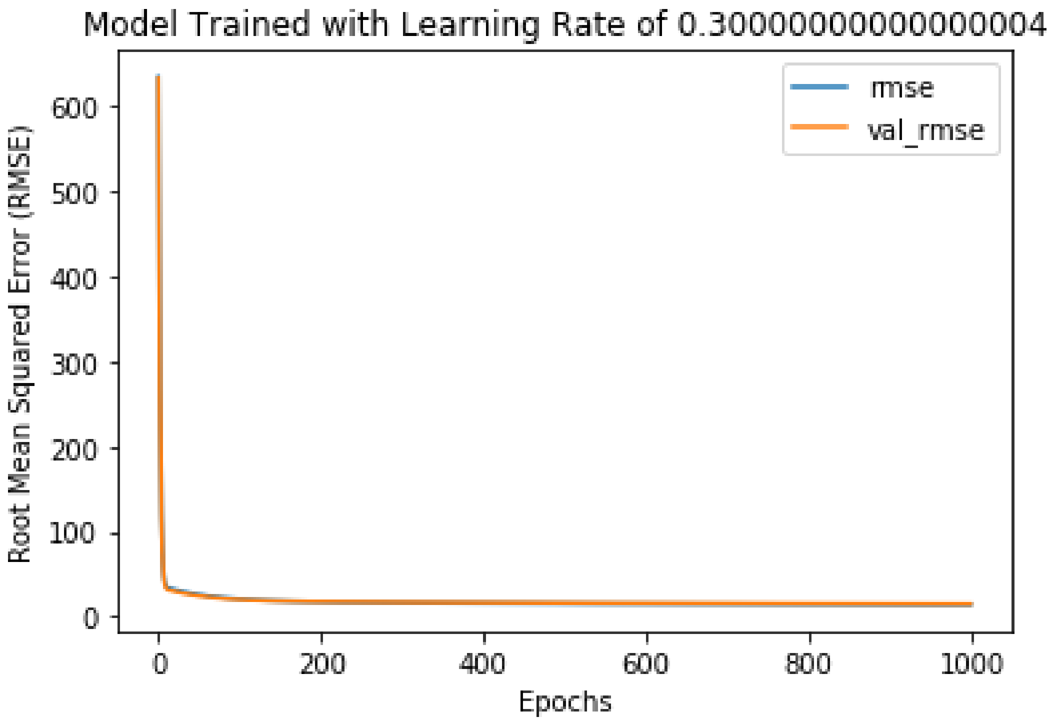
```
Model Trained with Learning Rate of 0.7000000000000001
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        5.179923
mse      104.984511
r_sqr      0.965446
rmse      10.246195
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        5.386293
val_mse      120.189784
val_r_sqr      0.949562
val_rmse      10.963110
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         777.561693
rmse         27.884793
mae          14.325743
r_sqr         0.804767
st_mse        0.000993
st_rmse       0.031512
st_mae        0.014685
st_r_sqr      0.950056
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.8
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.580505
mse      152.342796
r_sqr      0.947733
rmse      12.342722
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
```

```
val_mae        7.899153
val_mse      215.759098
val_r_sqr      0.940280
val_rmse      14.688741
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         99.381783
rmse         9.969041
mae          5.665376
r_sqr        0.967669
st_mse       0.000953
st_rmse      0.030868
st_mae       0.015169
st_r_sqr     0.937715
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 0.9
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.368586
mse      154.791285
r_sqr      0.953307
rmse      12.441515
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        6.191001
val_mse      124.999111
val_r_sqr      0.952075
val_rmse      11.180300
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         353.851397
rmse         18.810938
mae          10.752358
r_sqr         0.879272
st_mse        0.000580
st_rmse       0.024089
```

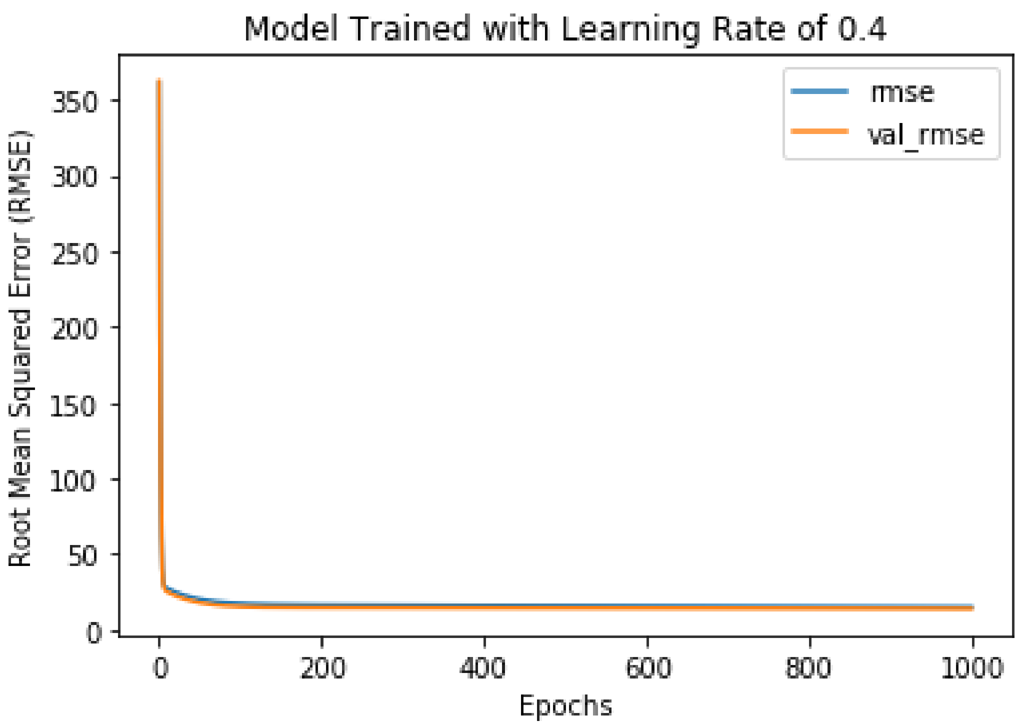
```
st_mae        0.012475
st_r_sqr      0.960218
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Learning Rate of 1.0
--------------------------------------------------------------------------------
--------------------
Final Training results
mae        6.446859
mse      141.742626
r_sqr      0.955973
rmse      11.905571
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation results
val_mae        7.205870
val_mse      183.563748
val_r_sqr      0.938170
val_rmse      13.548570
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse        173.410666
rmse        13.168548
mae          7.166303
r_sqr        0.939510
st_mse       0.000876
st_rmse      0.029593
st_mae       0.015108
st_r_sqr     0.938673
Name: 0, dtype: float64
================================================================================
====================
```


Model Trained with Learning Rate of 0.1
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
80
70
60
50
40
30
20
10
0
200
400
600
800
1000

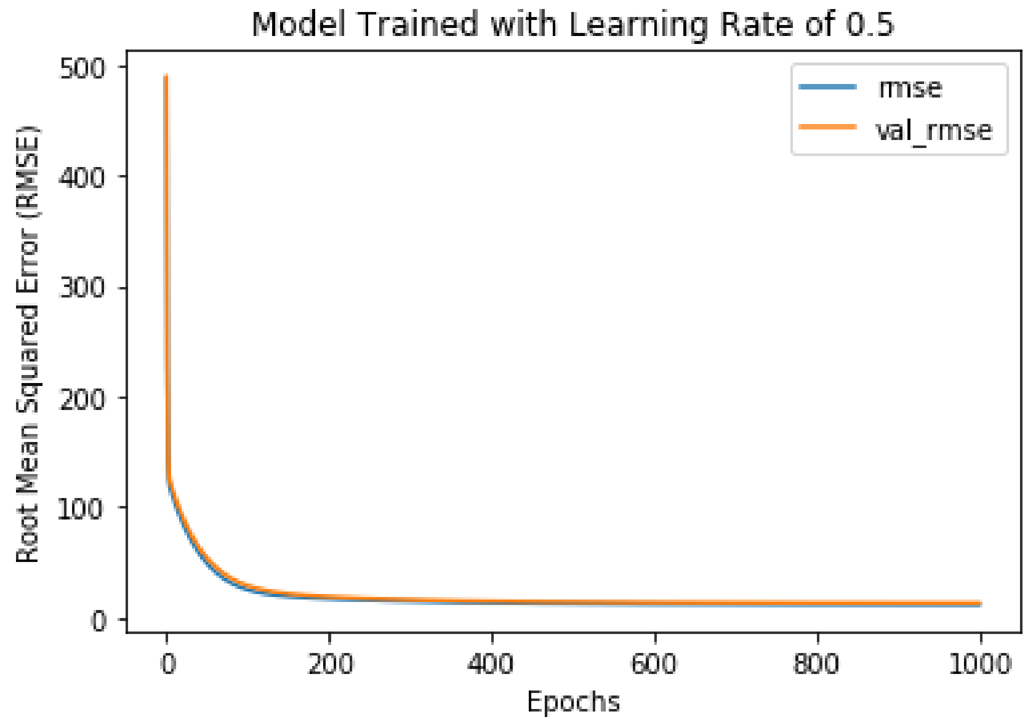

Model Trained with Learning Rate of 0.2
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
100
80
60
40
20
0
200
400
600
800
1000

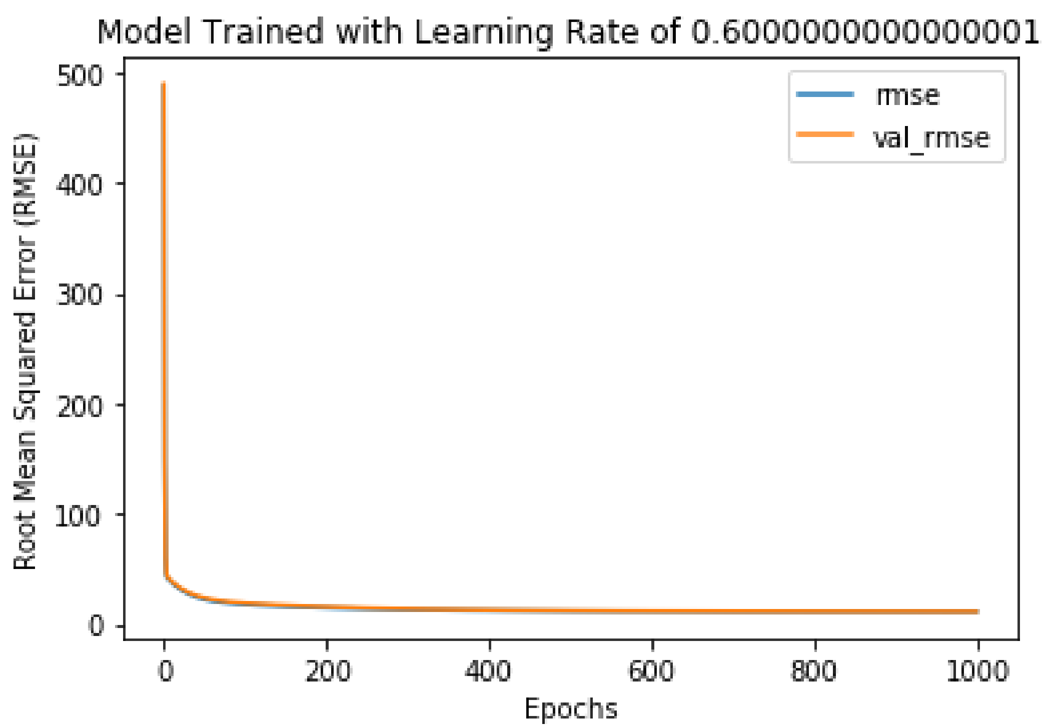


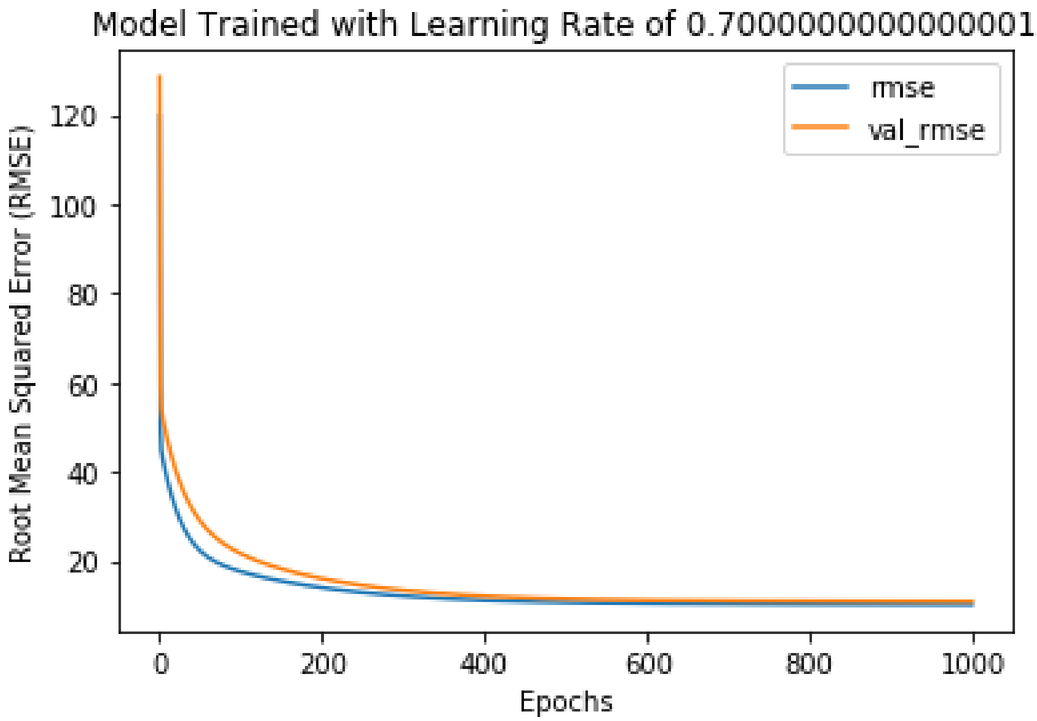
Model Trained with Learning Rate of 0.30000000000000004
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
600
0
200
400
600
800
1000
Epochs

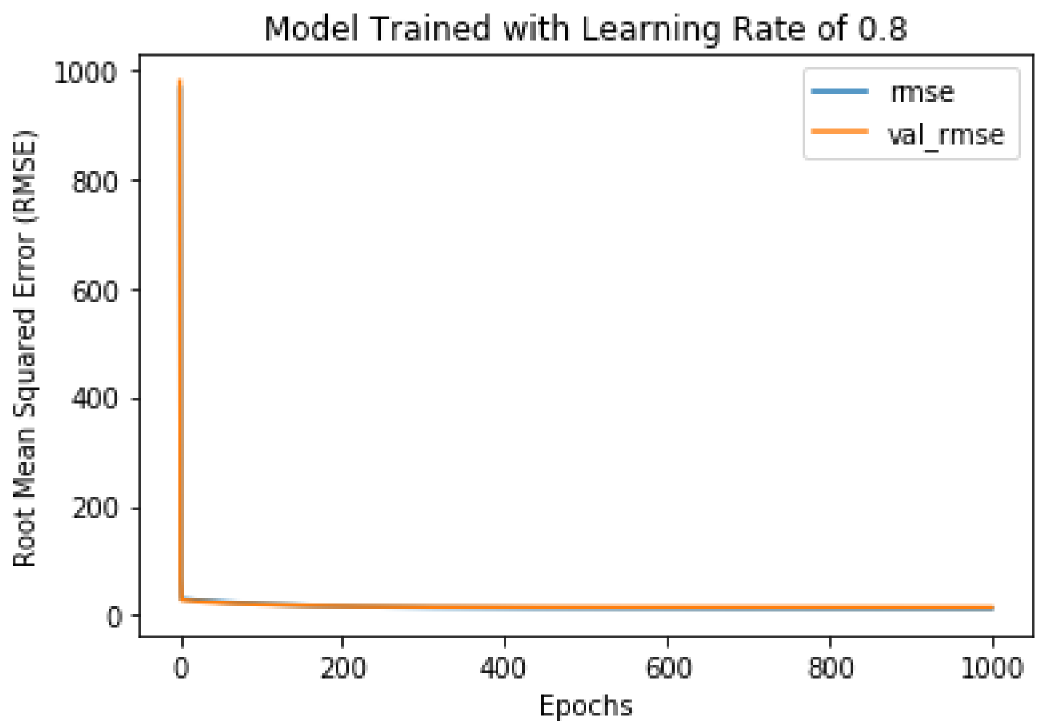


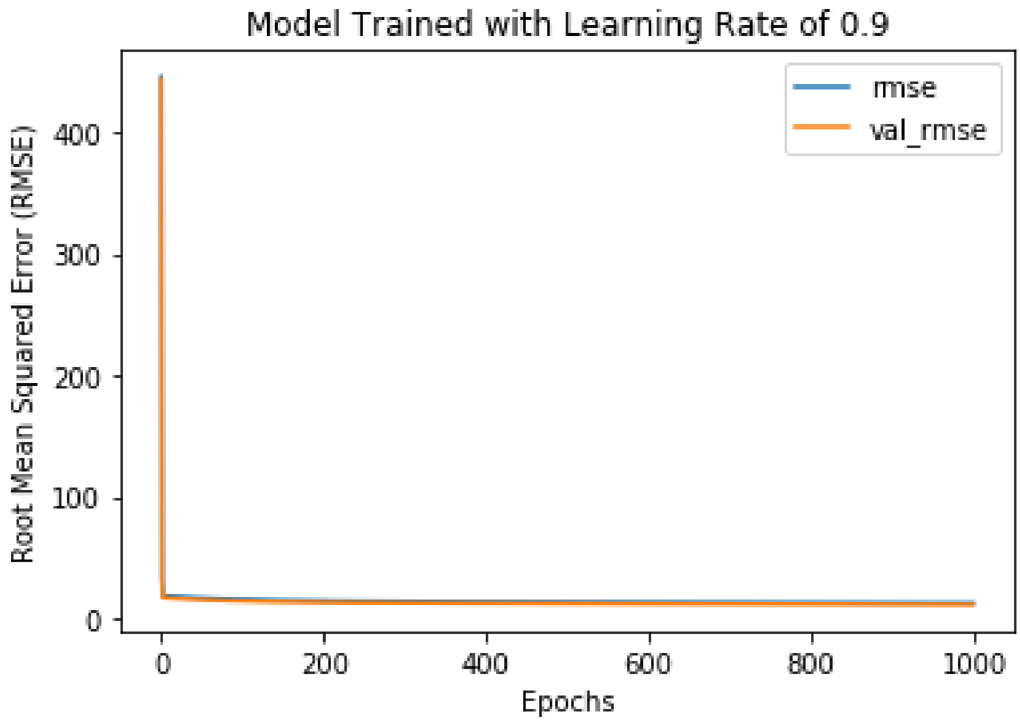
Model Trained with Learning Rate of 0.4
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
50
100
150
200
250
300
350
0
200
400
600
800
1000
Epochs

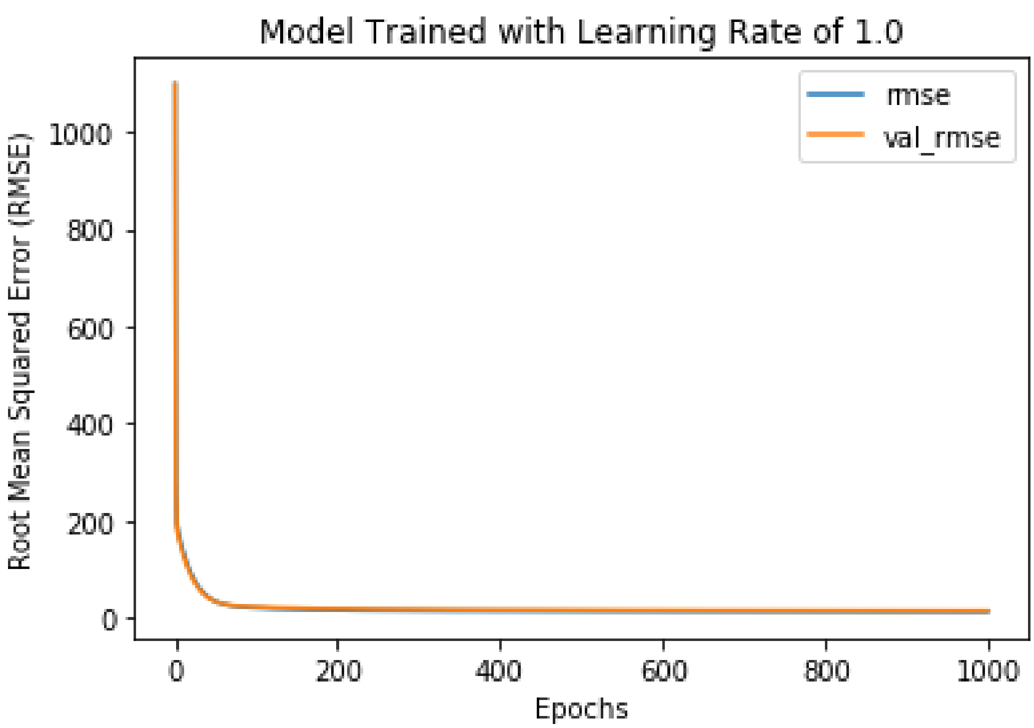

Model Trained with Learning Rate of 0.5
Root Mean Squared Error (RMSE)
500
400
300
200
100
0
0
200
400
600
800
1000
Epochs
rmse
val_rmse


Model Trained with Learning Rate of 0.6000000000000001
Root Mean Squared Error (RMSE)
500
400
300
200
100
0
0
200
400
600
800
1000
Epochs
rmse
val_rmse


Model Trained with Learning Rate of 0.7000000000000001
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
0
200
400
600
800
1000
20
40
60
80
100
120


Model Trained with Learning Rate of 0.8
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
0
200
400
600
800
1000


Model Trained with Learning Rate of 0.9
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 1.0
rmse
val_rmse
Root Mean Squared Error (RMSE)
1000
800
600
400
200
0
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

[15]:
```
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
    )
```

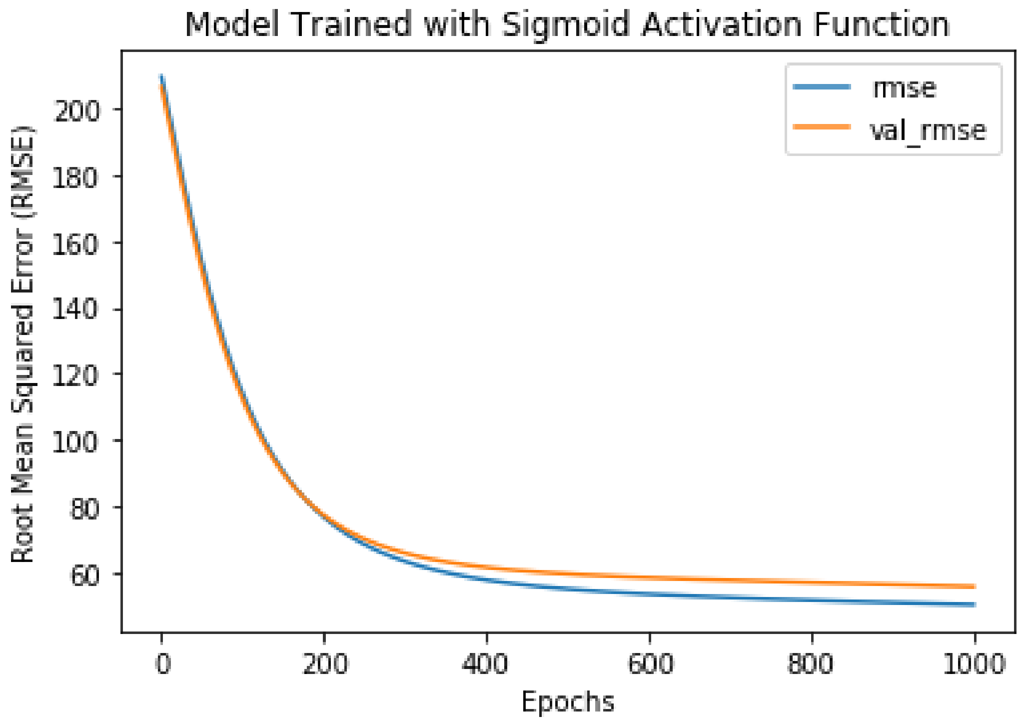
[16]:
```
for a in activations:
    print(f"Model Trained with {a.capitalize()} Activation Function",␣
↪end=f"\n{'-'*100}\n")

    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)")
```

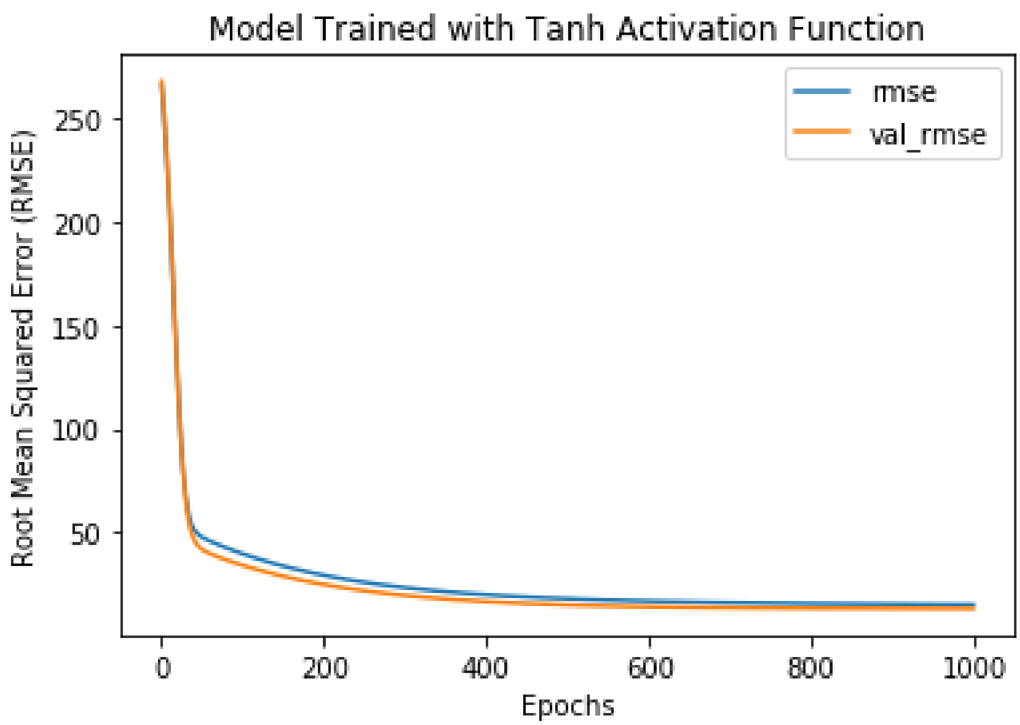
```
Model Trained with Sigmoid Activation Function
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        38.514325
mse      2535.839810
r_sqr       0.099437
rmse       50.357123
Name: 999, dtype: float64
```

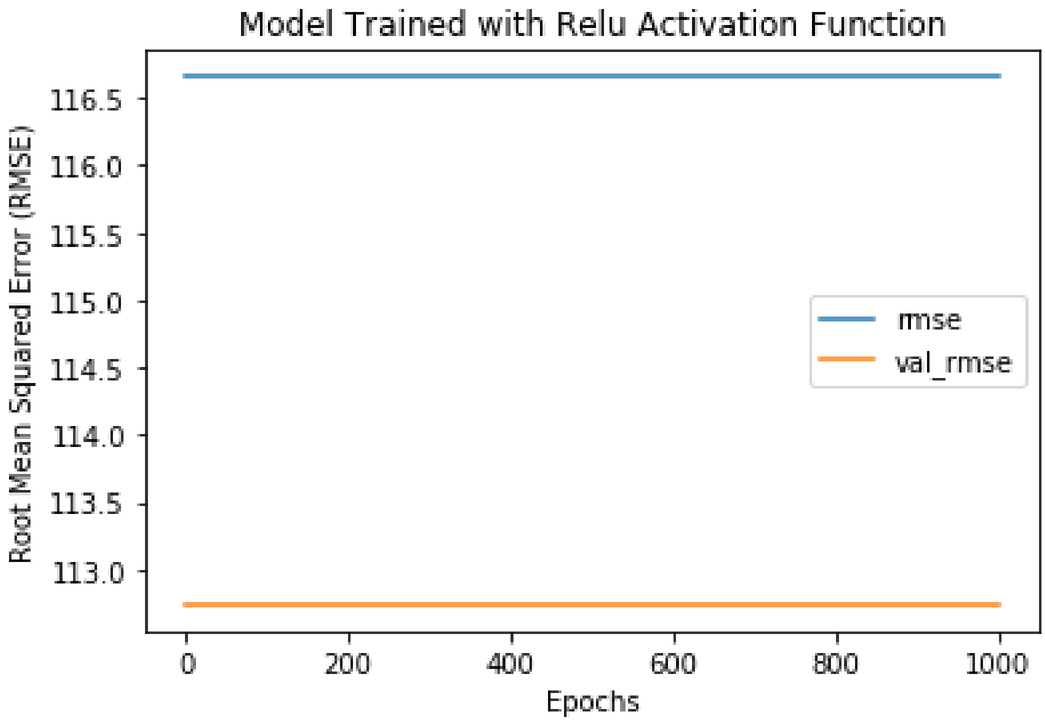
```
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        40.178245
val_mse      3115.092457
val_r_sqr       0.116448
val_rmse       55.813013
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         3205.065470
rmse          56.613298
mae           41.138513
r_sqr          0.081623
st_mse         0.010746
st_rmse        0.103662
st_mae         0.082285
st_r_sqr       0.050099
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Tanh Activation Function
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.854036
mse      230.046785
r_sqr      0.928153
rmse      15.167293
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.966150
val_mse      175.057784
val_r_sqr      0.927396
val_rmse      13.230940
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         278.640078
rmse         16.692516
mae           9.667961
```

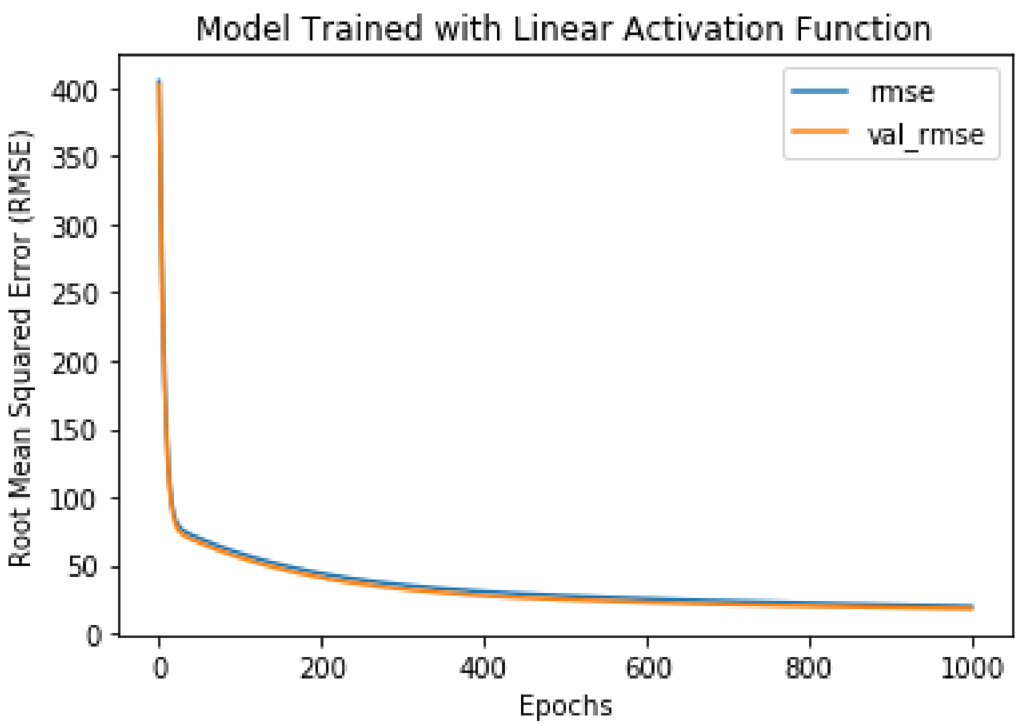
```
r_sqr         0.919583
st_mse        0.001965
st_rmse       0.044330
st_mae        0.029884
st_r_sqr      0.825110
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Relu Activation Function
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        100.179389
mse      13609.240910
r_sqr       -2.808559
rmse       116.658651
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae         99.653643
val_mse      12710.176655
val_r_sqr       -3.573111
val_rmse       112.739419
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         10807.651568
rmse          103.959856
mae            93.997537
r_sqr          -4.480235
st_mse          0.053553
st_rmse         0.231416
st_mae          0.199983
st_r_sqr       -2.949367
Name: 0, dtype: float64
================================================================================
====================



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

```
Final Training Results
mae       12.242387
mse      389.890987
r_sqr      0.868005
rmse      19.745657
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae       11.112784
val_mse      330.478029
val_r_sqr      0.895496
val_rmse      18.179055
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         496.884198
rmse         22.290899
mae          13.422733
r_sqr         0.856823
st_mse        0.001640
st_rmse       0.040497
st_mae        0.023602
st_r_sqr      0.854173
Name: 0, dtype: float64
================================================================================
====================
```


Model Trained with Sigmoid Activation Function
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
0
200
400
600
800
1000
60
80
100
120
140
160
180
200


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


Model Trained with Relu Activation Function
Root Mean Squared Error (RMSE)
116.5
116.0
115.5
115.0
114.5
114.0
113.5
113.0
0
200
400
600
800
1000
Epochs
rmse
val_rmse


Model Trained with Linear Activation Function
Root Mean Squared Error (RMSE)
400
350
300
250
200
150
100
50
0
0
200
400
600
800
1000
Epochs
rmse
val_rmse

- tanh is quite clearly the best performing activation function

Hidden Layers

[23]:
```
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
    )
```

[24]:
```
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         6.537448
mse       141.683314
r_sqr       0.953481
rmse       11.903080
```

```
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        6.708576
val_mse      143.885159
val_r_sqr      0.946122
val_rmse      11.995214
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         727.067186
rmse         26.964183
mae          15.432374
r_sqr         0.802375
st_mse        0.001012
st_rmse       0.031812
st_mae        0.017118
st_r_sqr      0.957242
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 2
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae       10.205298
mse      298.325655
r_sqr      0.904494
rmse      17.272106
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.514489
val_mse      230.076913
val_r_sqr      0.907843
val_rmse      15.168286
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         510.334071
rmse         22.590575
```

```
mae          13.883413
r_sqr         0.858922
st_mse        0.003538
st_rmse       0.059479
st_mae        0.027689
st_r_sqr      0.849668
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 3
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.705780
mse      184.419617
r_sqr      0.939006
rmse      13.580118
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.703655
val_mse      217.514316
val_r_sqr      0.925730
val_rmse      14.748367
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         865.644607
rmse         29.421839
mae          20.407450
r_sqr         0.752135
st_mse        0.001487
st_rmse       0.038567
st_mae        0.021021
st_r_sqr      0.933960
Name: 0, dtype: float64
================================================================================
====================



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

```
--------------------
Final Training Results
mae       11.749247
mse      349.502596
r_sqr      0.892044
rmse      18.694989
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.944671
val_mse      190.068800
val_r_sqr      0.905531
val_rmse      13.786544
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         428.229721
rmse         20.693712
mae          14.541870
r_sqr         0.885195
st_mse        0.002819
st_rmse       0.053096
st_mae        0.028576
st_r_sqr      0.868651
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 5
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae       10.956204
mse      323.253611
r_sqr      0.891739
rmse      17.979255
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae       11.432056
val_mse      330.366397
val_r_sqr      0.903258
val_rmse      18.175984
```

```
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         583.194248
rmse         24.149415
mae          15.859290
r_sqr         0.812844
st_mse        0.002409
st_rmse       0.049082
st_mae        0.025178
st_r_sqr      0.881684
Name: 0, dtype: float64
================================================================================
====================
```

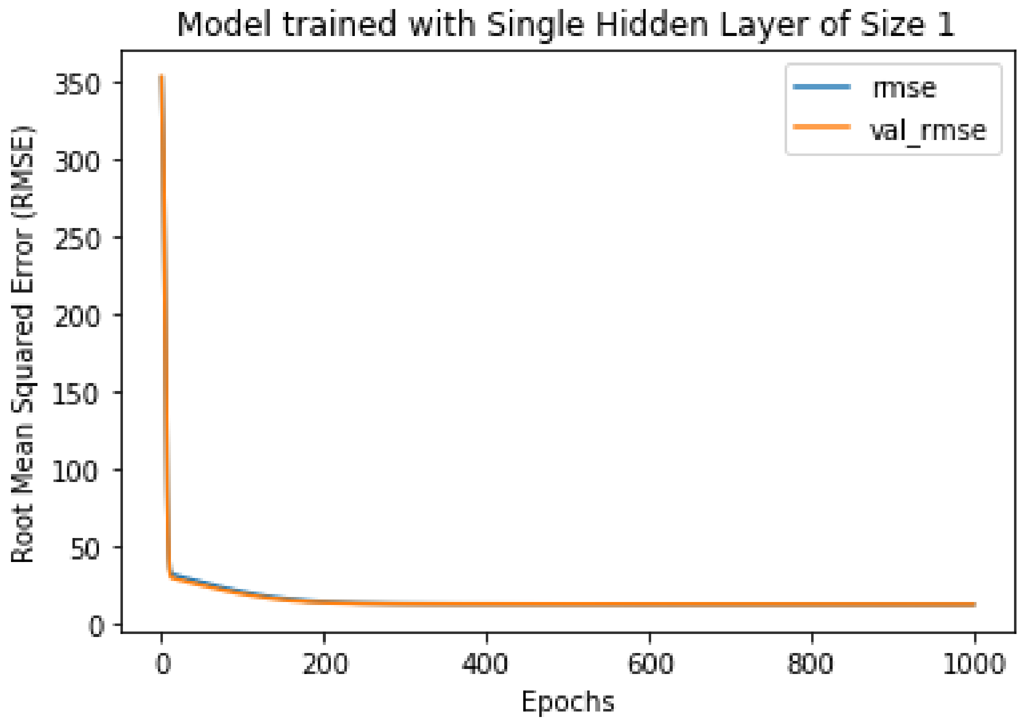
```
Model trained with Single Hidden Layer of Size 6
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.474634
mse      217.399014
r_sqr      0.917411
rmse      14.744457
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.297501
val_mse      301.599923
val_r_sqr      0.904126
val_rmse      17.366632
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         427.127333
rmse         20.667059
mae          10.664607
r_sqr         0.903359
st_mse        0.001479
st_rmse       0.038454
st_mae        0.023261
st_r_sqr      0.896758
Name: 0, dtype: float64
================================================================================
```

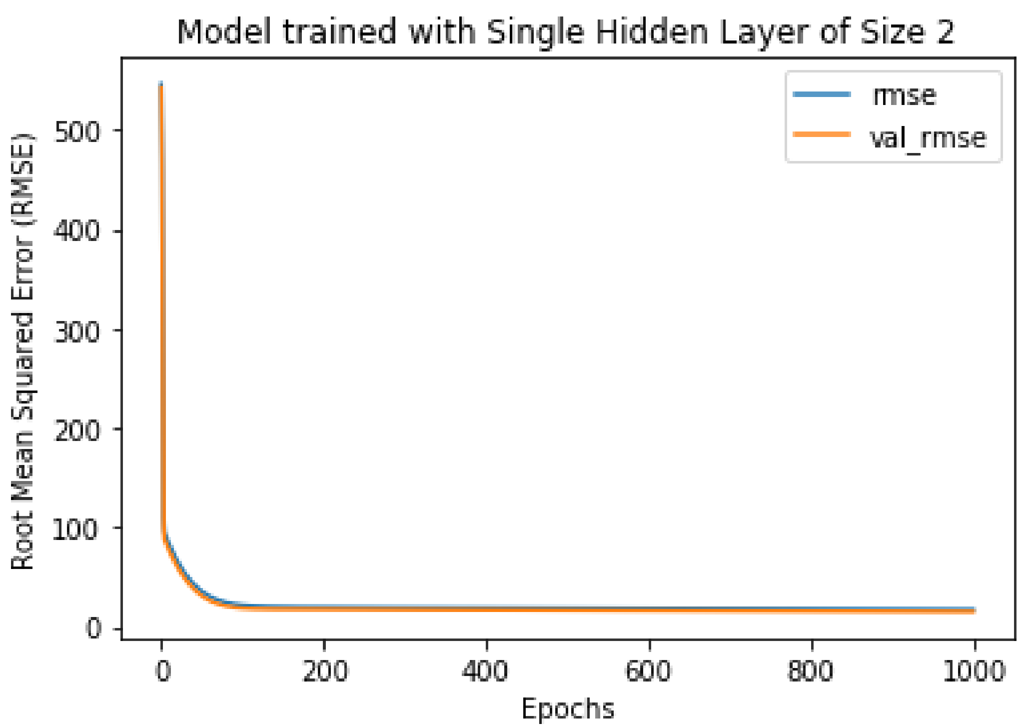
```
====================


Model trained with Single Hidden Layer of Size 7
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.878499
mse      185.417026
r_sqr      0.942938
rmse      13.616792
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.807907
val_mse      220.582351
val_r_sqr      0.922844
val_rmse      14.852015
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         644.390302
rmse         25.384844
mae          15.620218
r_sqr         0.776858
st_mse        0.002876
st_rmse       0.053625
st_mae        0.027910
st_r_sqr      0.880527
Name: 0, dtype: float64
================================================================================
====================


Model trained with Single Hidden Layer of Size 8
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        9.326841
mse      261.368740
r_sqr      0.922007
rmse      16.166903
Name: 999, dtype: float64
--------------------------------------------------------------------------------
```

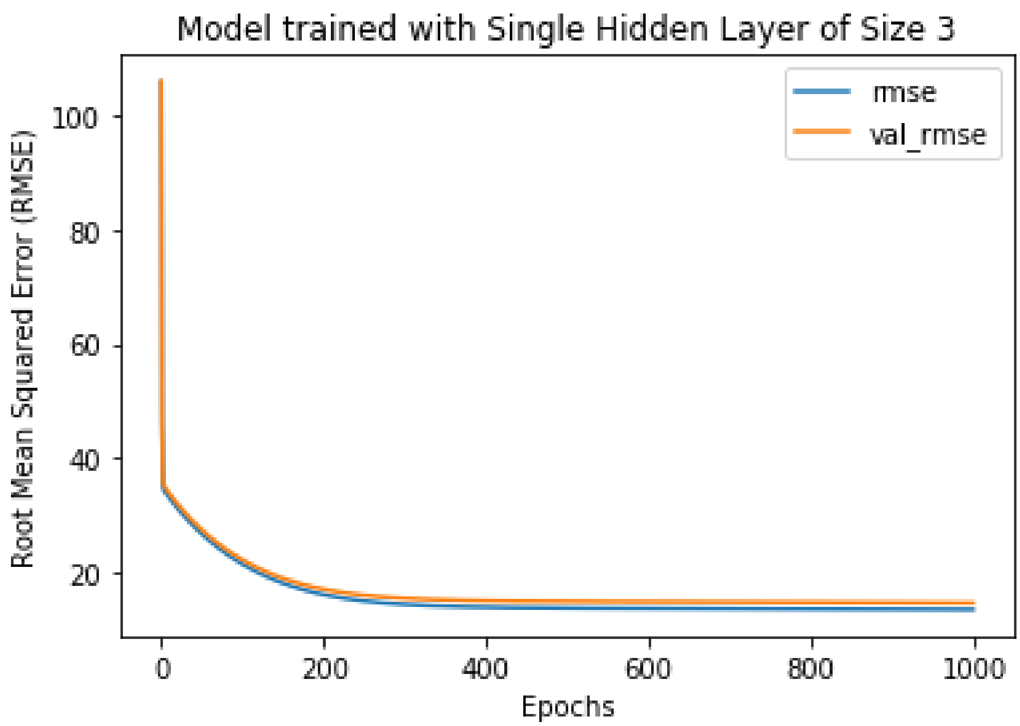
```
--------------------
Final Validation Results
val_mae        9.086521
val_mse      220.420668
val_r_sqr      0.929665
val_rmse      14.846571
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         459.131447
rmse         21.427353
mae          12.533972
r_sqr         0.799316
st_mse        0.003604
st_rmse       0.060036
st_mae        0.032903
st_r_sqr      0.816063
Name: 0, dtype: float64
================================================================================
====================
```

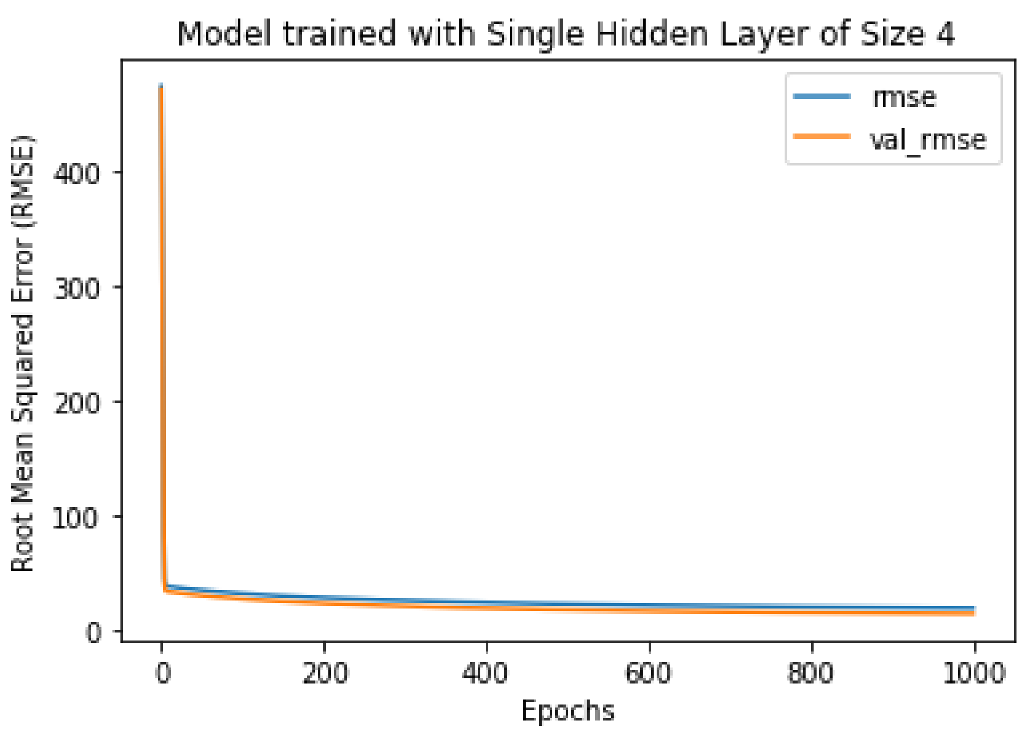
```
Model trained with Single Hidden Layer of Size 9
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.846222
mse      208.756315
r_sqr      0.933647
rmse      14.448402
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.027564
val_mse      162.109535
val_r_sqr      0.940558
val_rmse      12.732224
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         329.562140
rmse         18.153846
mae          12.064198
r_sqr         0.900492
```

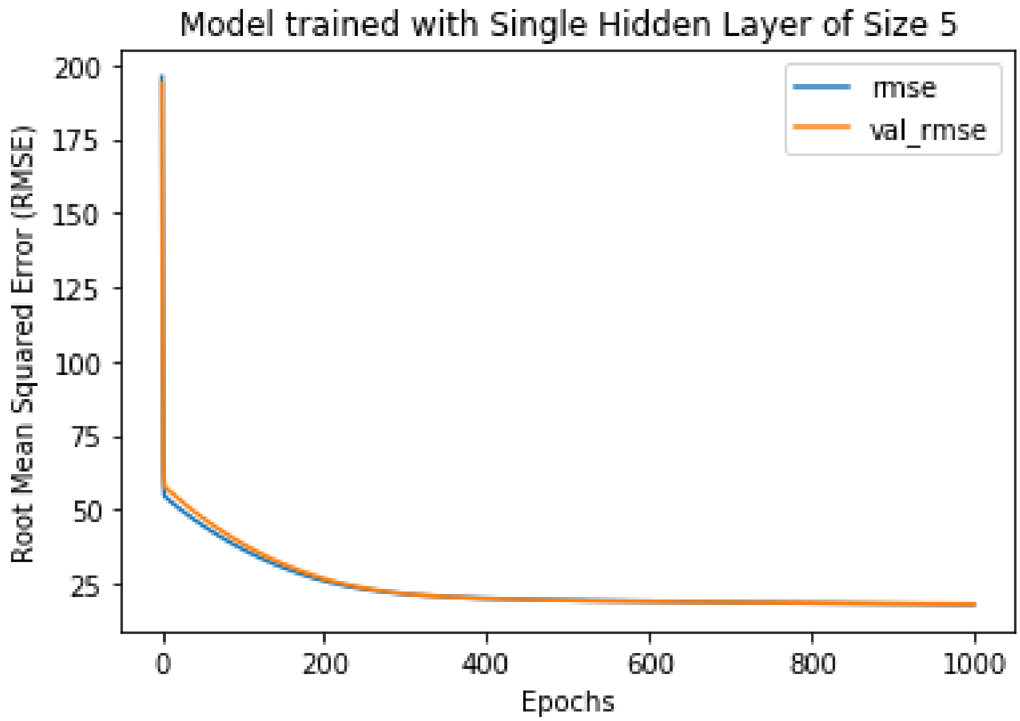
```
st_mse        0.001648
st_rmse       0.040592
st_mae        0.021139
st_r_sqr      0.900036
Name: 0, dtype: float64
================================================================================
====================



Model trained with Single Hidden Layer of Size 10
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.088137
mse      202.259429
r_sqr      0.930396
rmse      14.221794
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.665697
val_mse      191.862063
val_r_sqr      0.951003
val_rmse      13.851428
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         178.686358
rmse         13.367362
mae           8.343952
r_sqr         0.936293
st_mse        0.001677
st_rmse       0.040955
st_mae        0.022716
st_r_sqr      0.880053
Name: 0, dtype: float64
================================================================================
====================
```

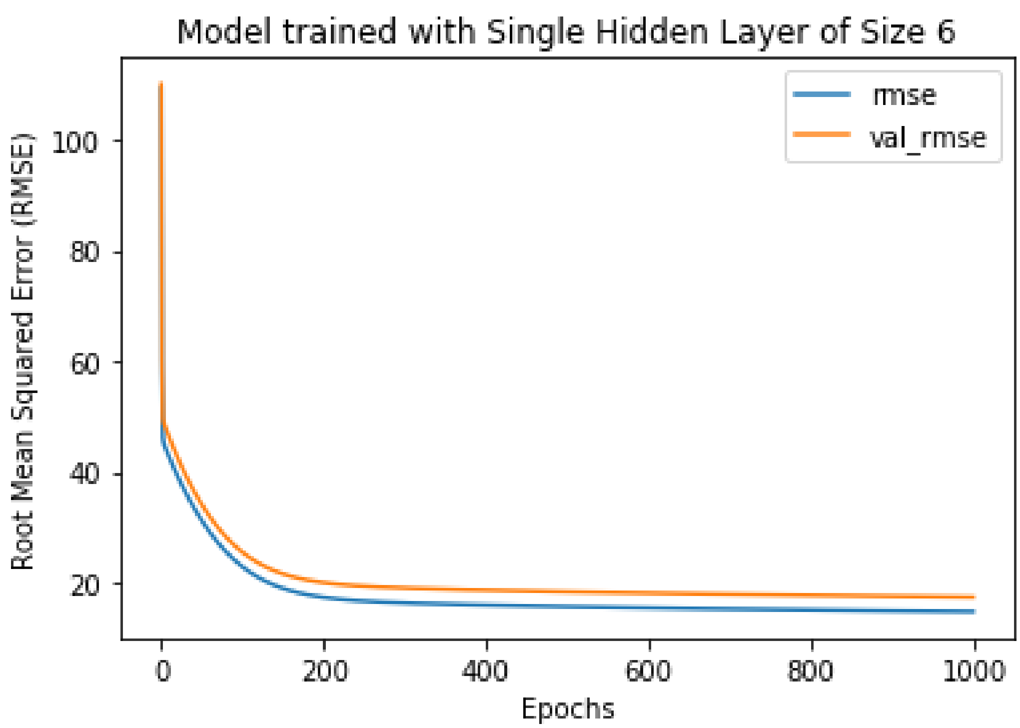

Model trained with Single Hidden Layer of Size 1
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
50
100
150
200
250
300
350
0
200
400
600
800
1000
Epochs

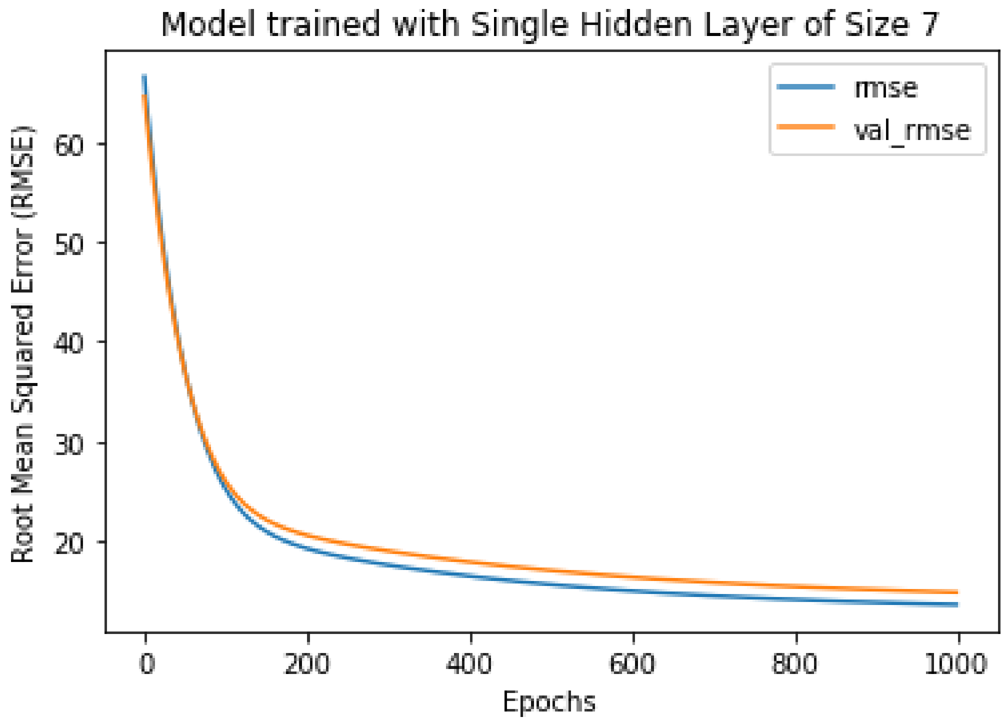

Model trained with Single Hidden Layer of Size 2
rmse
val_rmse
Root Mean Squared Error (RMSE)
0
100
200
300
400
500
0
200
400
600
800
1000
Epochs

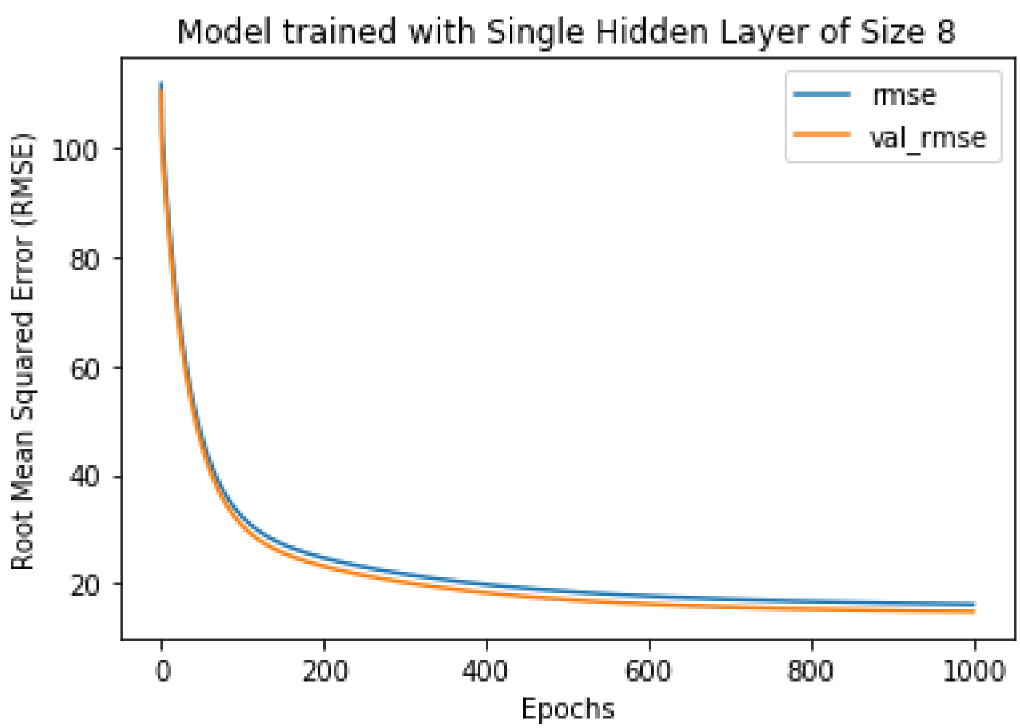

Model trained with Single Hidden Layer of Size 3
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
0
200
400
600
800
1000
20
40
60
80
100

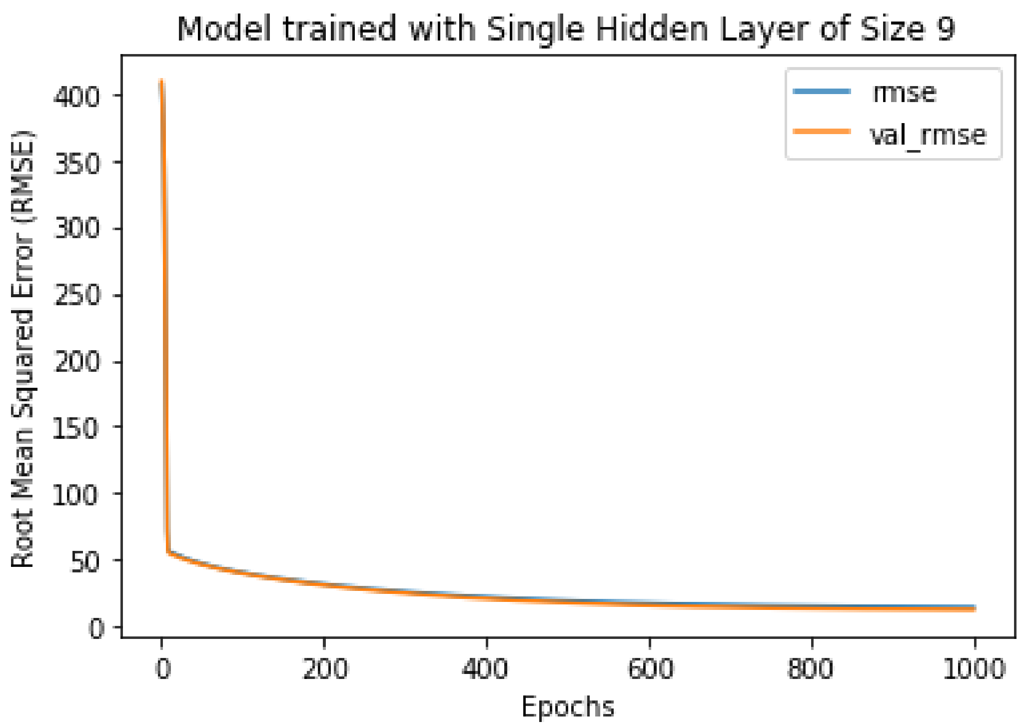

Model trained with Single Hidden Layer of Size 4
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
0
200
400
600
800
1000
0
100
200
300
400

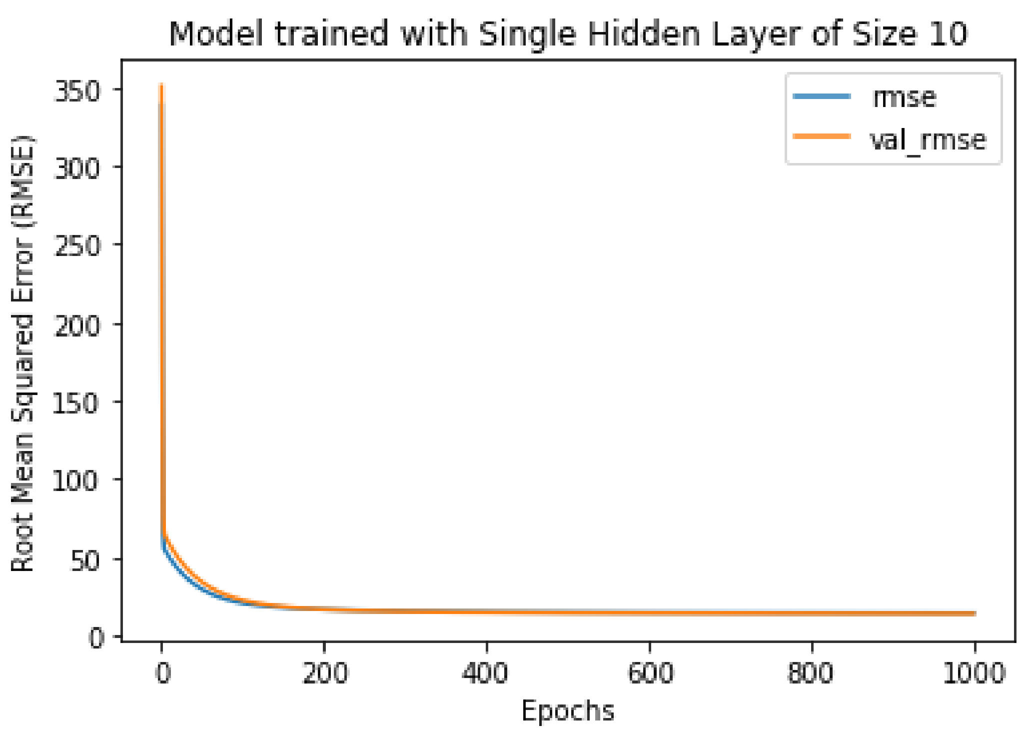

Model trained with Single Hidden Layer of Size 5
rmse
val_rmse
Root Mean Squared Error (RMSE)
200
175
150
125
100
75
50
25
0
200
400
600
800
1000
Epochs


Model trained with Single Hidden Layer of Size 6
rmse
val_rmse
Root Mean Squared Error (RMSE)
100
80
60
40
20
0
200
400
600
800
1000
Epochs


Model trained with Single Hidden Layer of Size 7
rmse
val_rmse
Root Mean Squared Error (RMSE)
20
30
40
50
60
0
200
400
600
800
1000
Epochs


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


Model trained with Single Hidden Layer of Size 9
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
0
50
100
150
200
250
300
350
400
0
200
400
600
800
1000


Model trained with Single Hidden Layer of Size 10
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
0
50
100
150
200
250
300
350
0
200
400
600
800
1000

```
[28]: 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
          )
```

```
[29]: 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        11.039224
mse       372.824255
r_sqr       0.892309
rmse       19.308658
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
```

```
val_mae        8.767553
val_mse      197.875779
val_r_sqr      0.918034
val_rmse      14.066833
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         327.410711
rmse         18.094494
mae          11.472224
r_sqr         0.877149
st_mse        0.006786
st_rmse       0.082379
st_mae        0.046483
st_r_sqr      0.703177
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 2)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae       14.960459
mse      528.579588
r_sqr      0.827110
rmse      22.990859
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae       13.888633
val_mse      505.515482
val_r_sqr      0.845042
val_rmse      22.483671
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         930.783480
rmse         30.508744
mae          22.436335
r_sqr         0.695578
st_mse        0.003913
st_rmse       0.062555
```

```
st_mae        0.035549
st_r_sqr      0.801065
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 3)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.733926
mse      228.135558
r_sqr      0.933564
rmse      15.104157
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.949196
val_mse      168.438061
val_r_sqr      0.932911
val_rmse      12.978369
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse        1792.981943
rmse         42.343617
mae          29.433090
r_sqr         0.327214
st_mse        0.002311
st_rmse       0.048068
st_mae        0.030563
st_r_sqr      0.865453
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 4)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae       10.329634
mse      297.055861
```

```
r_sqr      0.904130
rmse      17.235309
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae       10.720939
val_mse      273.030036
val_r_sqr      0.916427
val_rmse      16.523621
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         297.127479
rmse         17.237386
mae          11.082555
r_sqr         0.898321
st_mse        0.001642
st_rmse       0.040518
st_mae        0.022925
st_r_sqr      0.887155
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 5)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        6.650090
mse      154.343884
r_sqr      0.948926
rmse      12.423521
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.309605
val_mse      178.053204
val_r_sqr      0.945661
val_rmse      13.343658
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
```

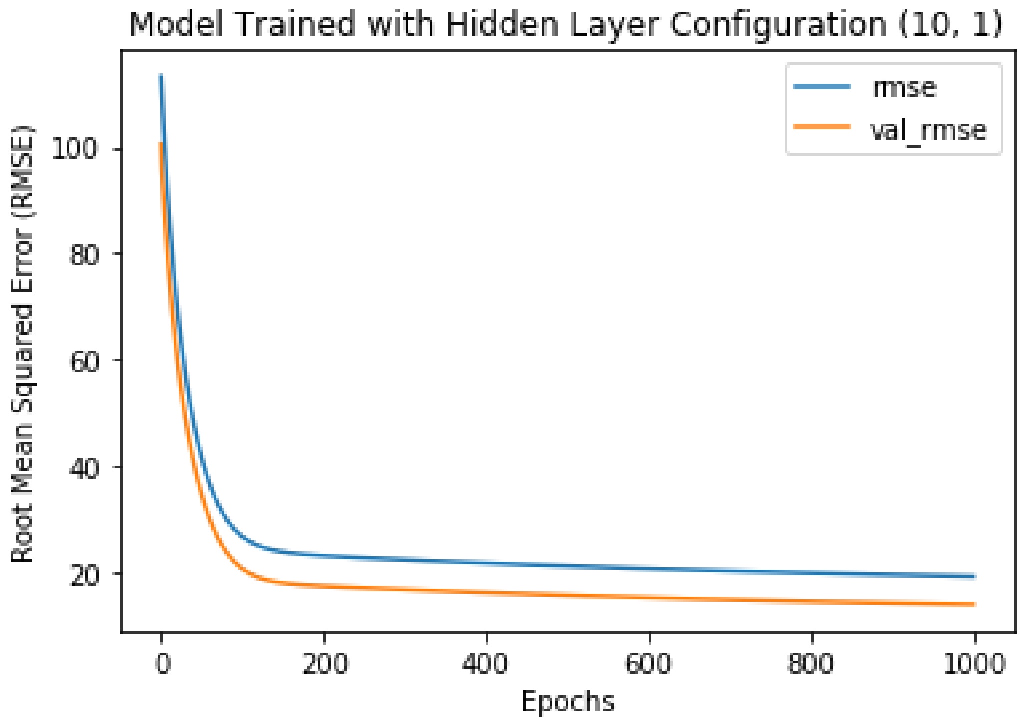
```
mse         301.371642
rmse         17.360059
mae          11.063077
r_sqr         0.904170
st_mse        0.001250
st_rmse       0.035358
st_mae        0.016522
st_r_sqr      0.931113
Name: 0, dtype: float64
================================================================================
====================
```

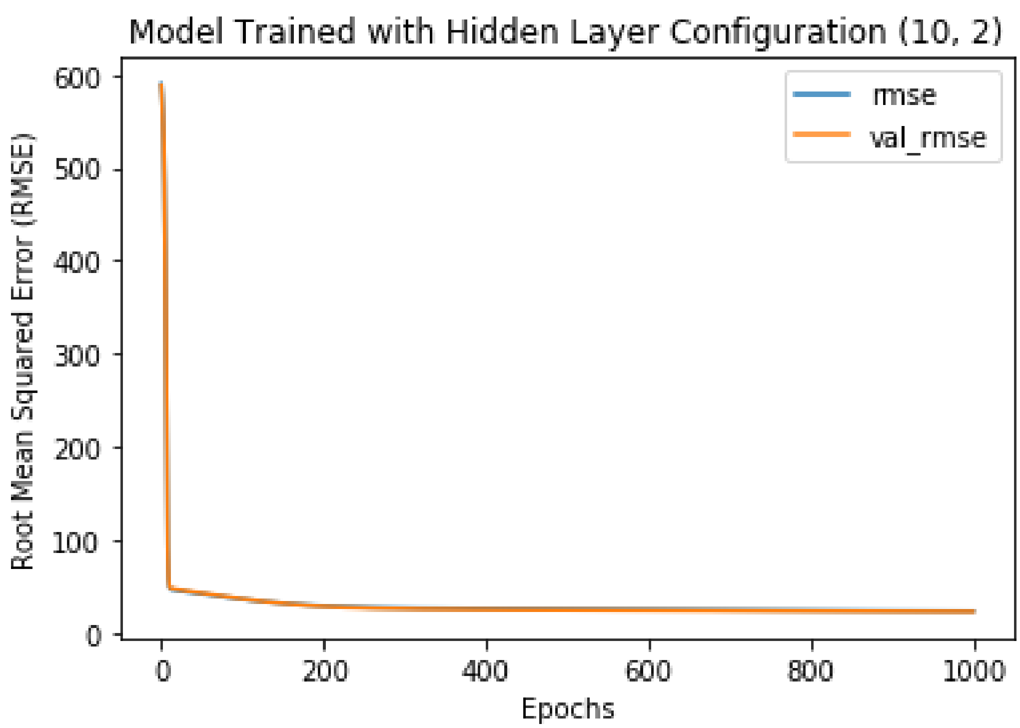
```
Model Trained with Hidden Layer Configuration (10, 6)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        9.374567
mse      238.607777
r_sqr      0.927337
rmse      15.446934
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.233305
val_mse      277.627647
val_r_sqr      0.925936
val_rmse      16.662162
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         2038.522945
rmse          45.150005
mae           32.770927
r_sqr         -0.095314
st_mse         0.002843
st_rmse        0.053322
st_mae         0.028133
st_r_sqr       0.887762
Name: 0, dtype: float64
================================================================================
====================
```

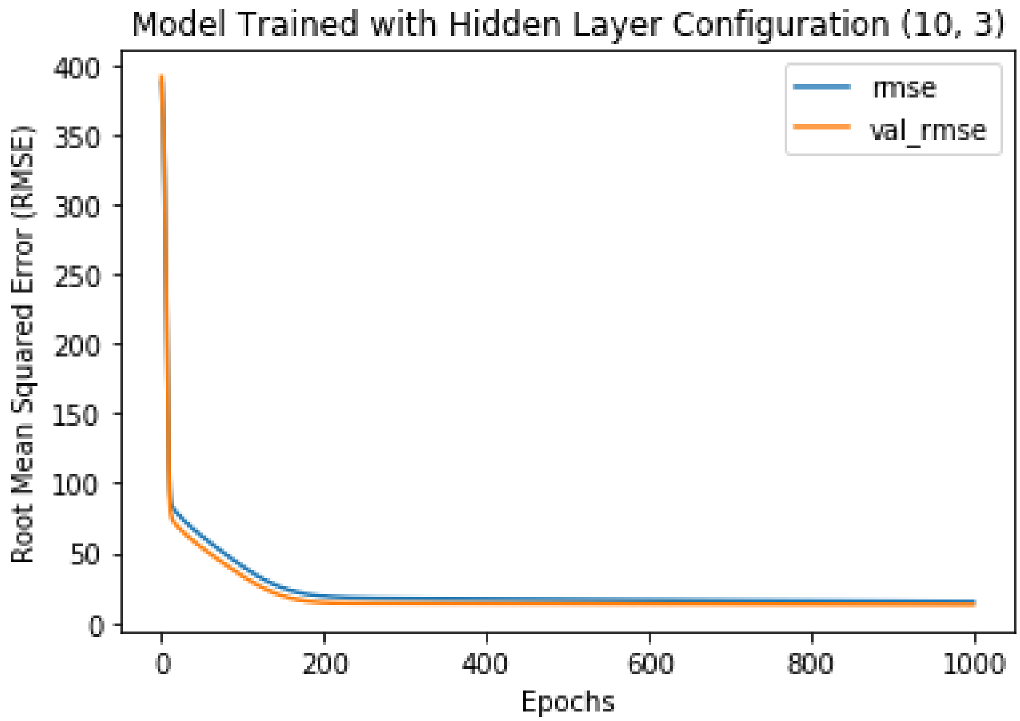
```
Model Trained with Hidden Layer Configuration (10, 7)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        6.788405
mse      155.399767
r_sqr      0.943998
rmse      12.465944
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.665292
val_mse      240.926042
val_r_sqr      0.936648
val_rmse      15.521792
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         505.007571
rmse         22.472373
mae          14.885314
r_sqr         0.849342
st_mse        0.001550
st_rmse       0.039370
st_mae        0.019088
st_r_sqr      0.919637
Name: 0, dtype: float64
================================================================================
====================


Model Trained with Hidden Layer Configuration (10, 8)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        9.593583
mse      264.748185
r_sqr      0.911685
rmse      16.271084
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.723533
val_mse      261.125326
```

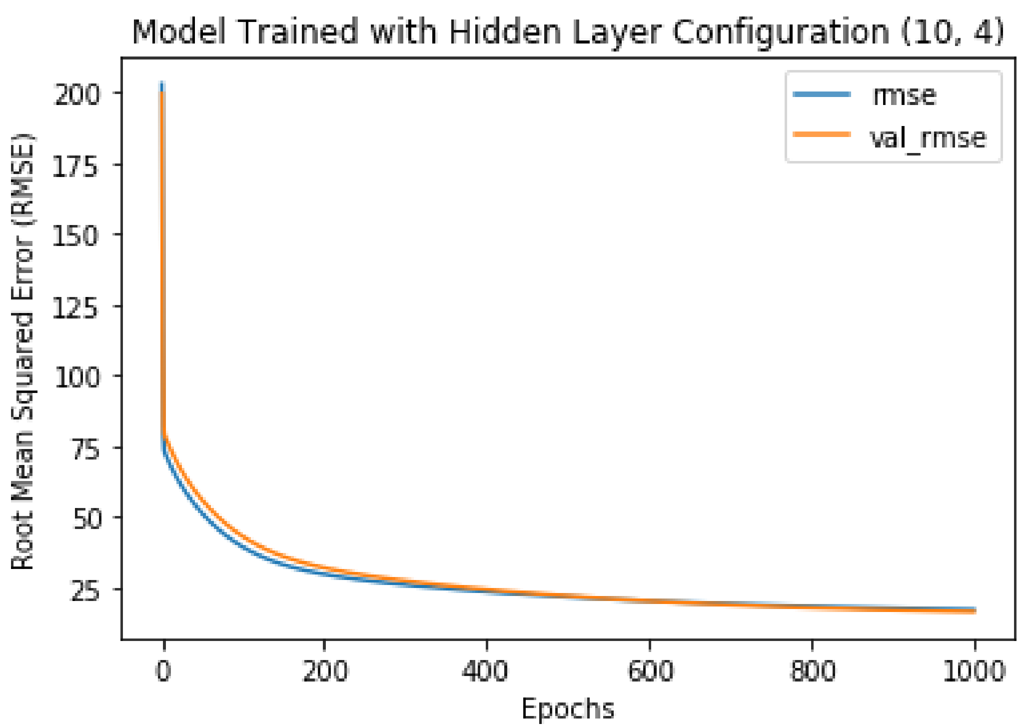
```
val_r_sqr      0.918077
val_rmse      16.159373
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         257.575178
rmse         16.049149
mae           9.836957
r_sqr         0.922273
st_mse        0.002280
st_rmse       0.047754
st_mae        0.024605
st_r_sqr      0.861724
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 9)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.164161
mse      176.956075
r_sqr      0.938995
rmse      13.302484
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.147647
val_mse      305.478118
val_r_sqr      0.910433
val_rmse      17.477932
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         470.744577
rmse         21.696649
mae          12.374243
r_sqr         0.858922
st_mse        0.005060
st_rmse       0.071134
st_mae        0.038185
st_r_sqr      0.818139
```

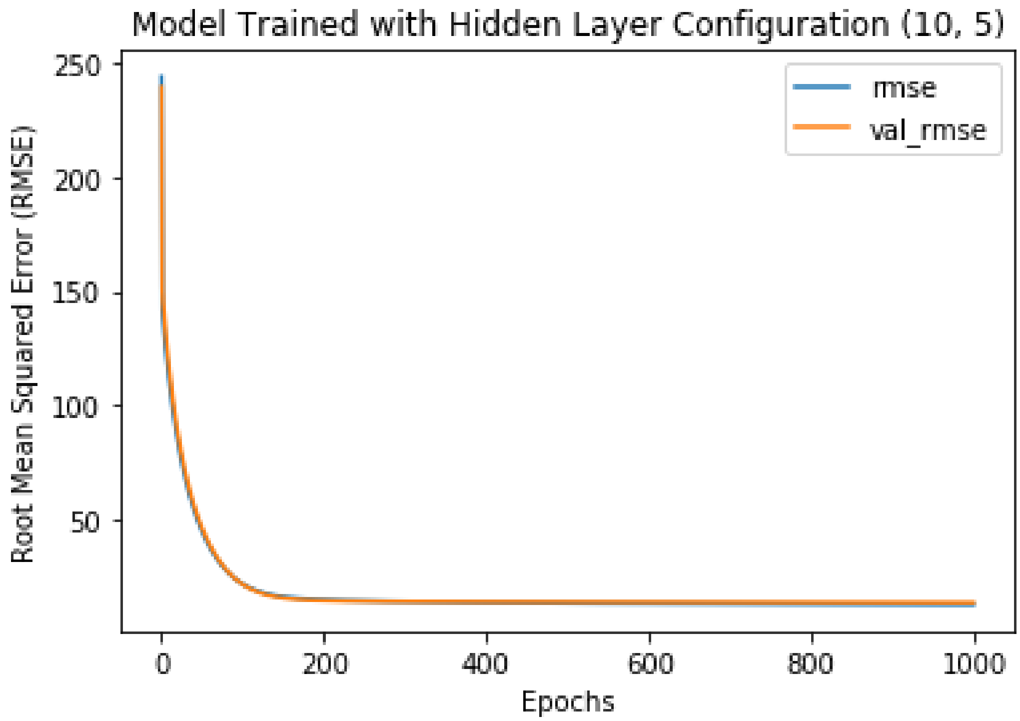
```
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.182189
mse      216.348942
r_sqr      0.923964
rmse      14.708805
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.491528
val_mse      295.626709
val_r_sqr      0.916262
val_rmse      17.193799
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         418.041829
rmse         20.446071
mae          10.285933
r_sqr         0.877130
st_mse        0.000812
st_rmse       0.028500
st_mae        0.015443
st_r_sqr      0.926349
Name: 0, dtype: float64
================================================================================
====================
```

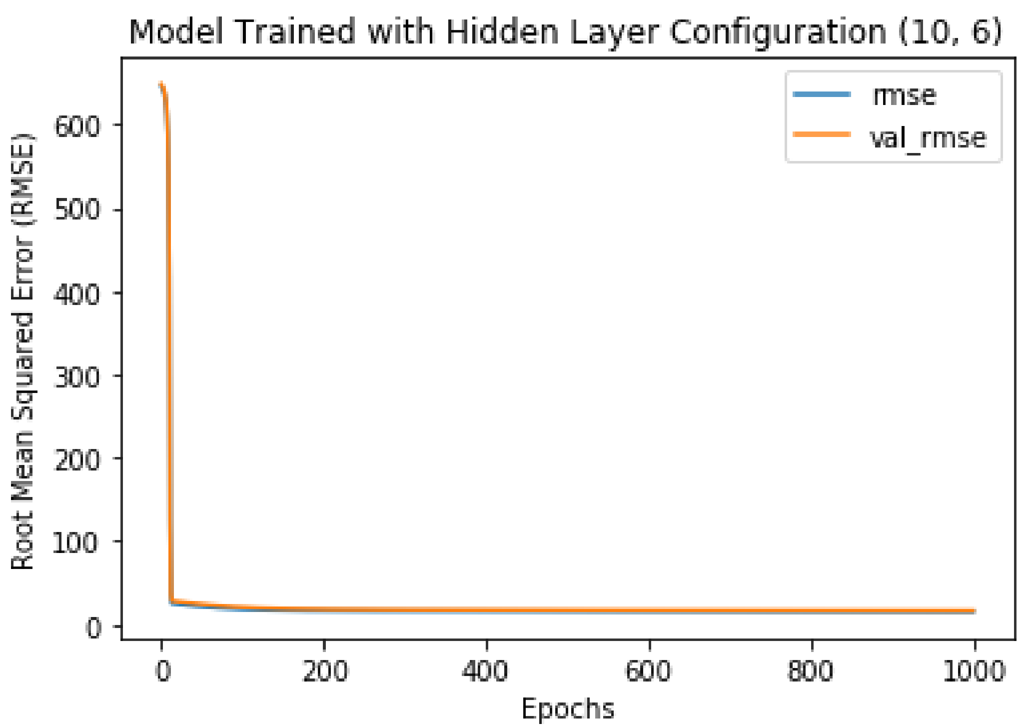

Model Trained with Hidden Layer Configuration (10, 1)
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
0
200
400
600
800
1000
20
40
60
80
100

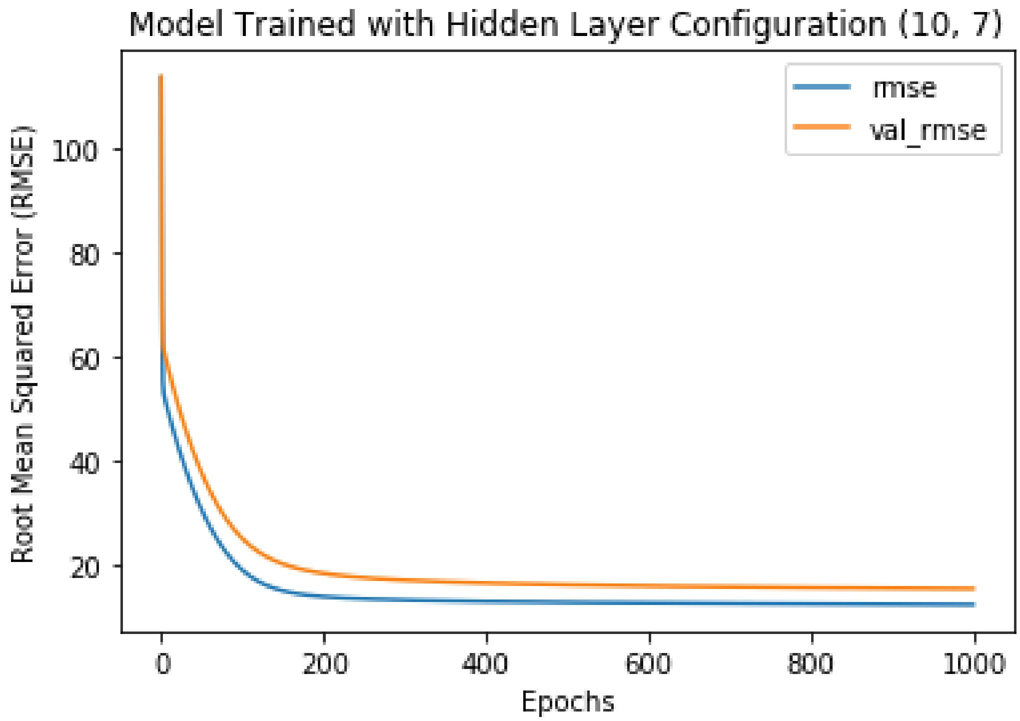

Model Trained with Hidden Layer Configuration (10, 2)
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
0
200
400
600
800
1000
0
100
200
300
400
500
600

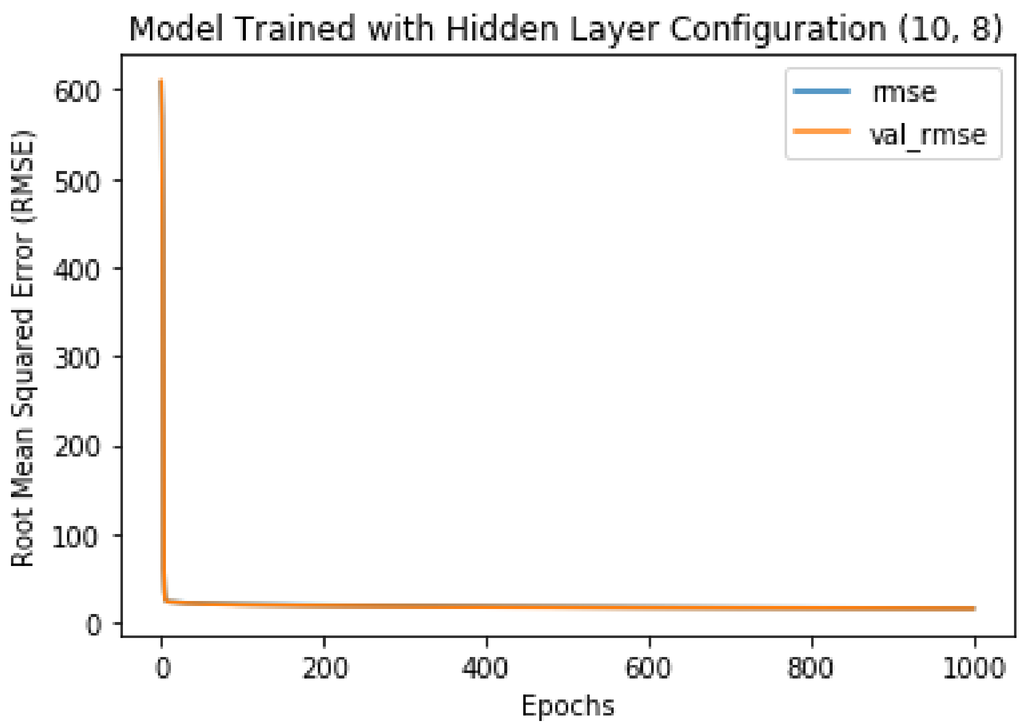

Model Trained with Hidden Layer Configuration (10, 3)
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
0
50
100
150
200
250
300
350
400
200
400
600
800
1000

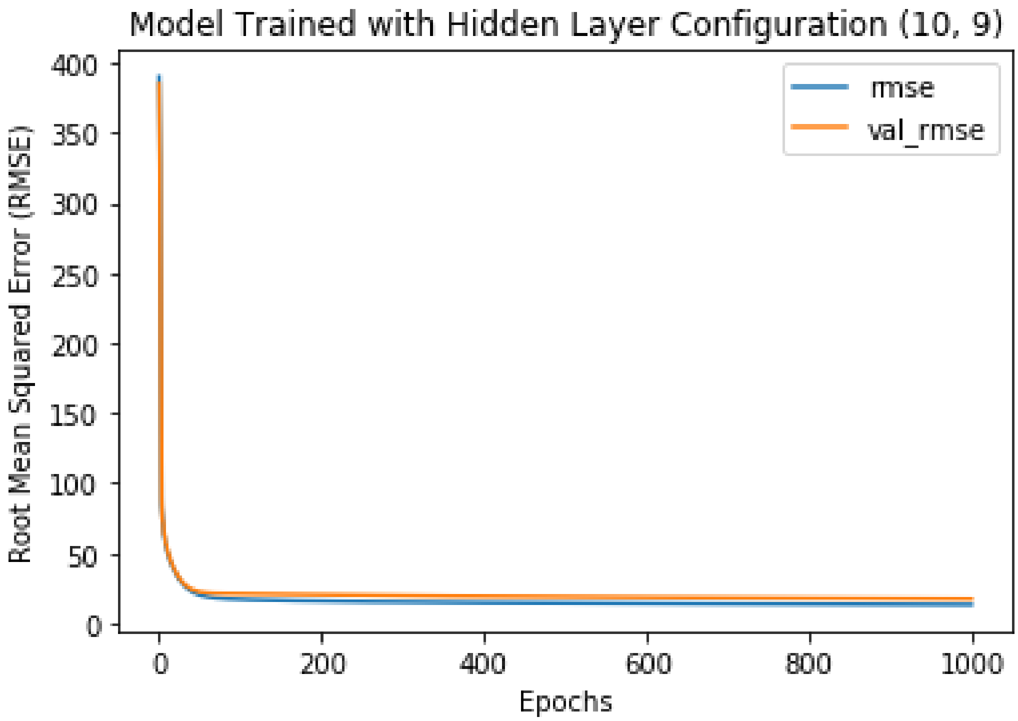

Model Trained with Hidden Layer Configuration (10, 4)
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
25
50
75
100
125
150
175
200
0
400
600
800
1000


Model Trained with Hidden Layer Configuration (10, 5)
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
0
200
400
600
800
1000
50
100
150
200
250

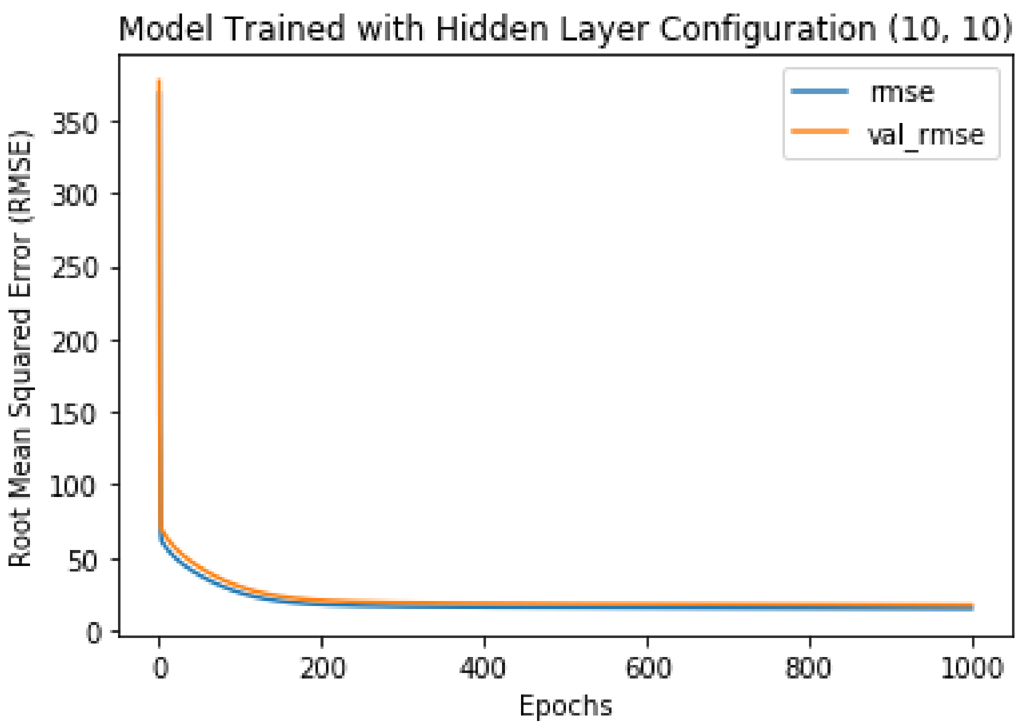

Model Trained with Hidden Layer Configuration (10, 6)
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
0
200
400
600
800
1000
0
100
200
300
400
500
600


Model Trained with Hidden Layer Configuration (10, 7)
rmse
val_rmse
Root Mean Squared Error (RMSE)
100
80
60
40
20
0
200
400
600
800
1000
Epochs


Model Trained with Hidden Layer Configuration (10, 8)
rmse
val_rmse
Root Mean Squared Error (RMSE)
600
500
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)
Epochs
0
200
400
600
800
1000
0
50
100
150
200
250
300
350
400



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

```
[32]: 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
    )
```

```
[33]: 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         40.675349
mse       3289.605540
r_sqr        0.000141
rmse        57.355083
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
```

```
val_mae        40.452444
val_mse      3191.763601
val_r_sqr      -0.000552
val_rmse       56.495695
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         2442.108068
rmse          49.417690
mae           36.684681
r_sqr         -0.005315
st_mse         0.024735
st_rmse        0.157273
st_mae         0.094491
st_r_sqr      -0.084921
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (2, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        9.205754
mse      279.591917
r_sqr      0.907125
rmse      16.721002
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae       10.477098
val_mse      355.636679
val_r_sqr      0.903670
val_rmse      18.858332
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         948.689127
rmse         30.800798
mae          19.482716
r_sqr         0.655228
st_mse        0.002617
st_rmse       0.051161
```

```
st_mae        0.023390
st_r_sqr      0.886004
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (3, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        29.776235
mse      1000.587635
r_sqr       0.651198
rmse       31.632067
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        30.532874
val_mse      1100.822412
val_r_sqr       0.716457
val_rmse       33.178644
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         1227.786313
rmse          35.039782
mae           30.576602
r_sqr          0.589698
st_mse         0.003323
st_rmse        0.057648
st_mae         0.050783
st_r_sqr       0.658818
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (4, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        9.533718
mse      299.505396
```

```
r_sqr      0.901843
rmse      17.306224
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.655035
val_mse      241.227884
val_r_sqr      0.908075
val_rmse      15.531513
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         545.763014
rmse         23.361571
mae          12.153721
r_sqr         0.852945
st_mse        0.001061
st_rmse       0.032575
st_mae        0.018374
st_r_sqr      0.911921
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (5, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.727363
mse      216.310404
r_sqr      0.928333
rmse      14.707495
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        9.774685
val_mse      322.415189
val_r_sqr      0.907832
val_rmse      17.955924
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
```

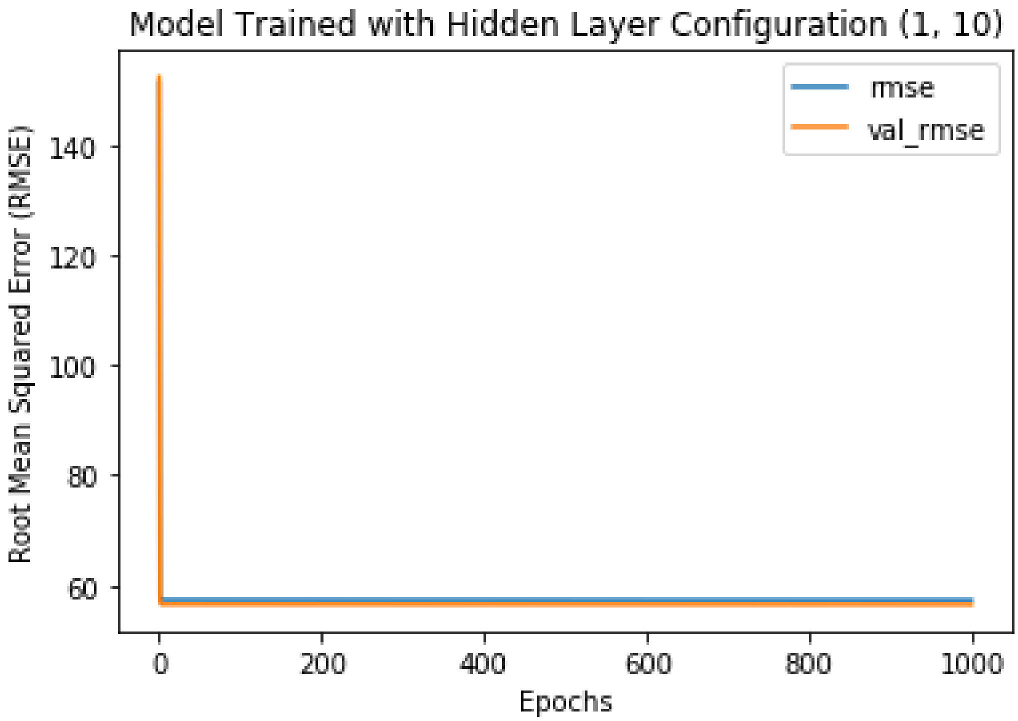
```
mse         487.227778
rmse         22.073237
mae          14.595240
r_sqr         0.834330
st_mse        0.002324
st_rmse       0.048203
st_mae        0.023174
st_r_sqr      0.878759
Name: 0, dtype: float64
================================================================================
====================
```

```
Model Trained with Hidden Layer Configuration (6, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        6.903358
mse      159.462782
r_sqr      0.952458
rmse      12.627857
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        6.846378
val_mse      169.339608
val_r_sqr      0.950305
val_rmse      13.013055
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         2610.194233
rmse          51.090060
mae           35.145469
r_sqr         -0.311430
st_mse         0.002085
st_rmse        0.045664
st_mae         0.022646
st_r_sqr       0.922044
Name: 0, dtype: float64
================================================================================
====================
```

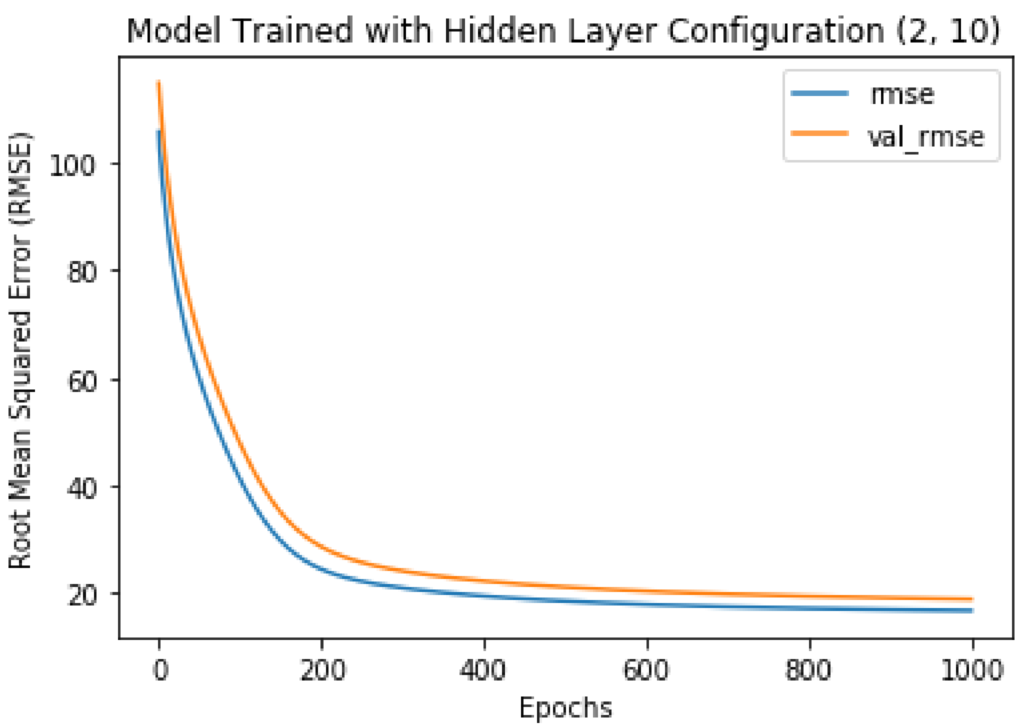
```
Model Trained with Hidden Layer Configuration (7, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        7.294836
mse      166.755323
r_sqr      0.948493
rmse      12.913378
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.466487
val_mse      188.788827
val_r_sqr      0.936254
val_rmse      13.740045
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         246.147520
rmse         15.689089
mae           7.769502
r_sqr         0.912560
st_mse        0.001219
st_rmse       0.034917
st_mae        0.019874
st_r_sqr      0.866350
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (8, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        6.053548
mse      124.757734
r_sqr      0.961100
rmse      11.169500
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        5.847862
val_mse      102.754911
```

```
val_r_sqr      0.961203
val_rmse      10.136810
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         2323.555325
rmse          48.203271
mae           34.414942
r_sqr          0.279108
st_mse         0.002943
st_rmse        0.054247
st_mae         0.036393
st_r_sqr       0.857782
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (9, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.399457
mse      218.070847
r_sqr      0.933390
rmse      14.767222
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        7.389402
val_mse      157.882046
val_r_sqr      0.941587
val_rmse      12.565112
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         706.222259
rmse         26.574843
mae          16.900812
r_sqr         0.761802
st_mse        0.001696
st_rmse       0.041186
st_mae        0.021709
st_r_sqr      0.910816
```

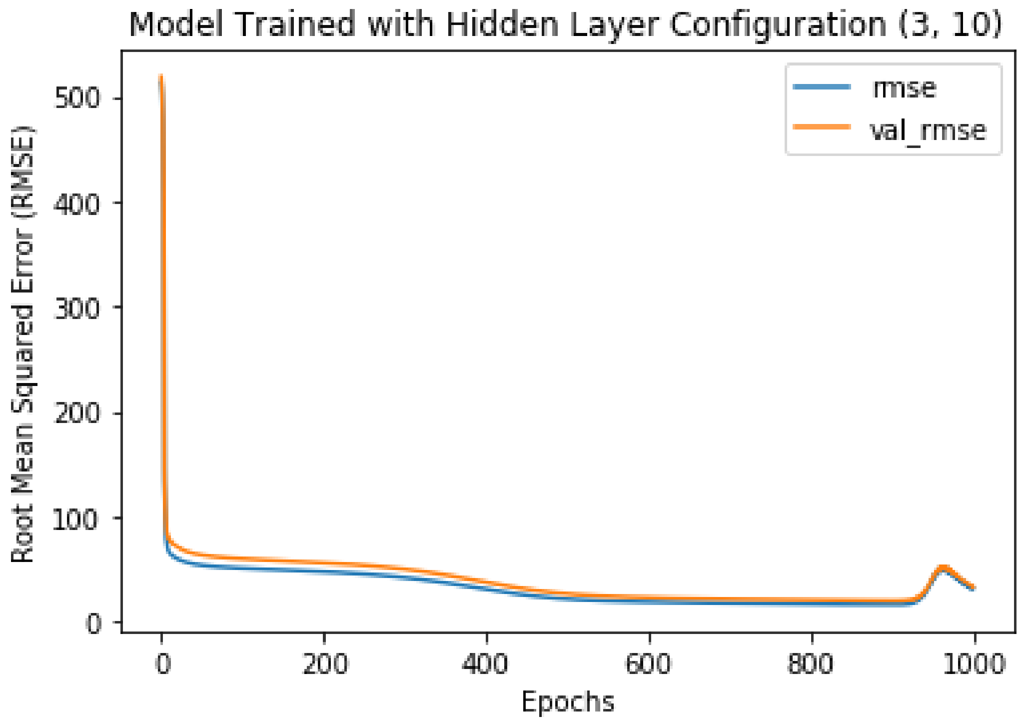
```
Name: 0, dtype: float64
================================================================================
====================



Model Trained with Hidden Layer Configuration (10, 10)
--------------------------------------------------------------------------------
--------------------
Final Training Results
mae        8.578838
mse      254.414124
r_sqr      0.913281
rmse      15.950364
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Final Validation Results
val_mae        8.812246
val_mse      340.880065
val_r_sqr      0.923956
val_rmse      18.462938
Name: 999, dtype: float64
--------------------------------------------------------------------------------
--------------------
Test Set Results
mse         225.380567
rmse         15.012680
mae           9.438959
r_sqr         0.896025
st_mse        0.005652
st_rmse       0.075180
st_mae        0.042396
st_r_sqr      0.721736
Name: 0, dtype: float64
================================================================================
====================
```

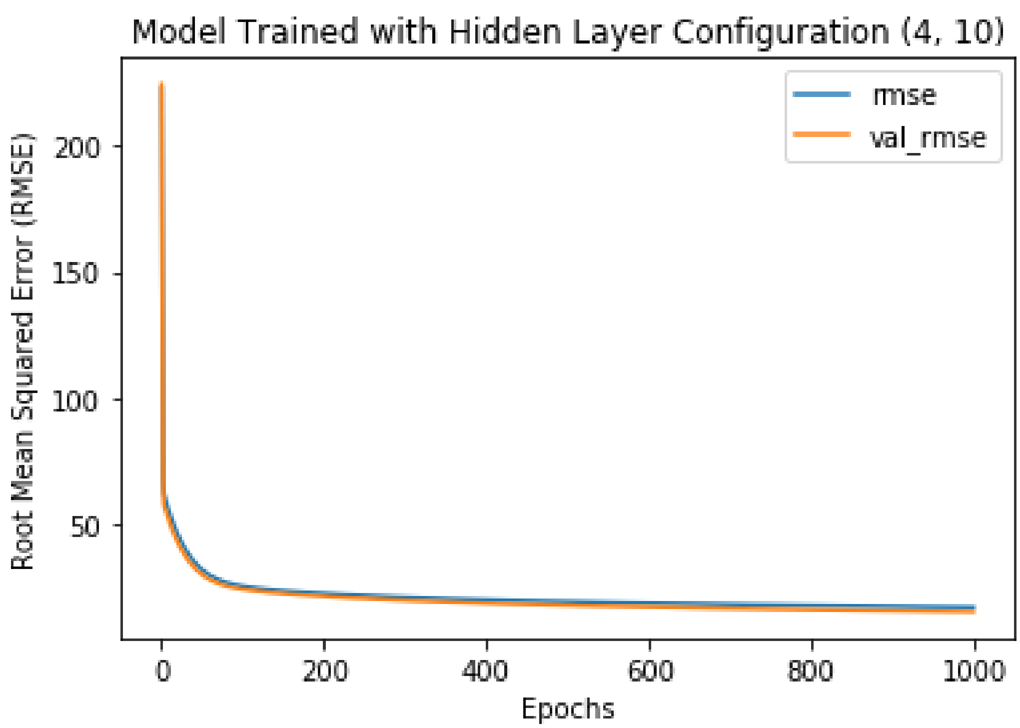


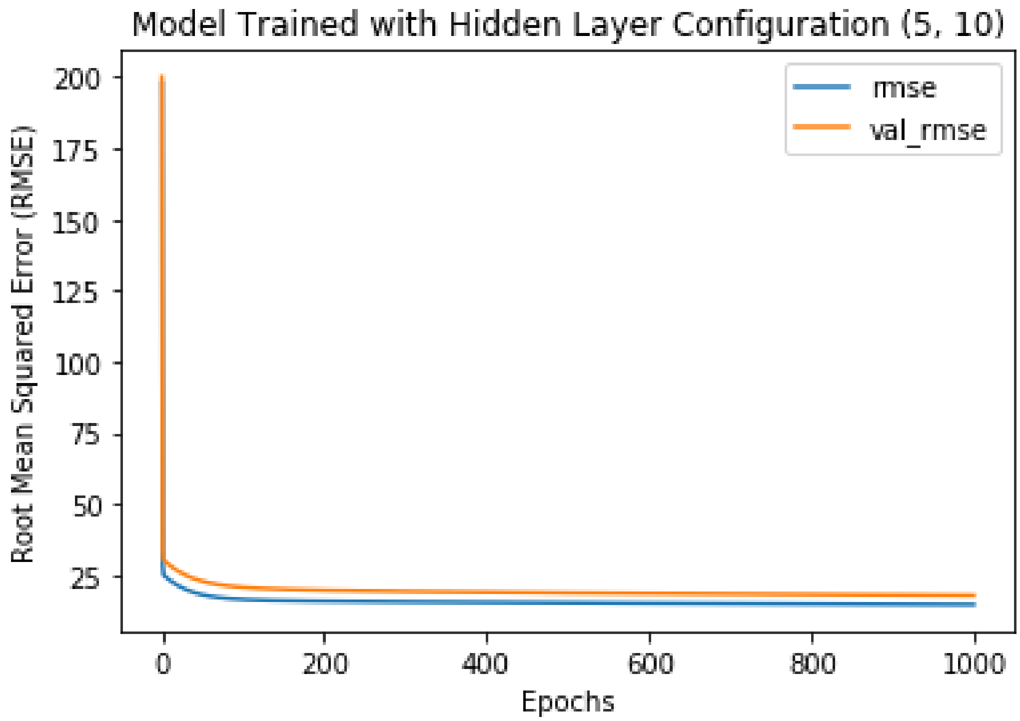
Model Trained with Hidden Layer Configuration (1, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
140
120
100
80
60
0
200
400
600
800
1000
Epochs

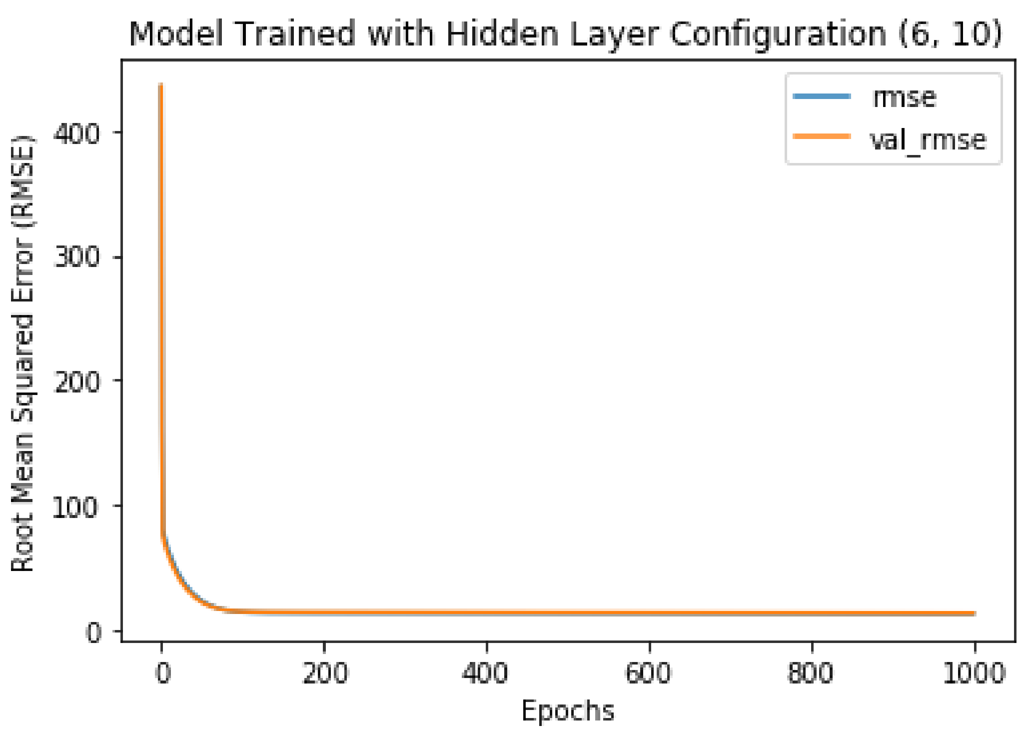


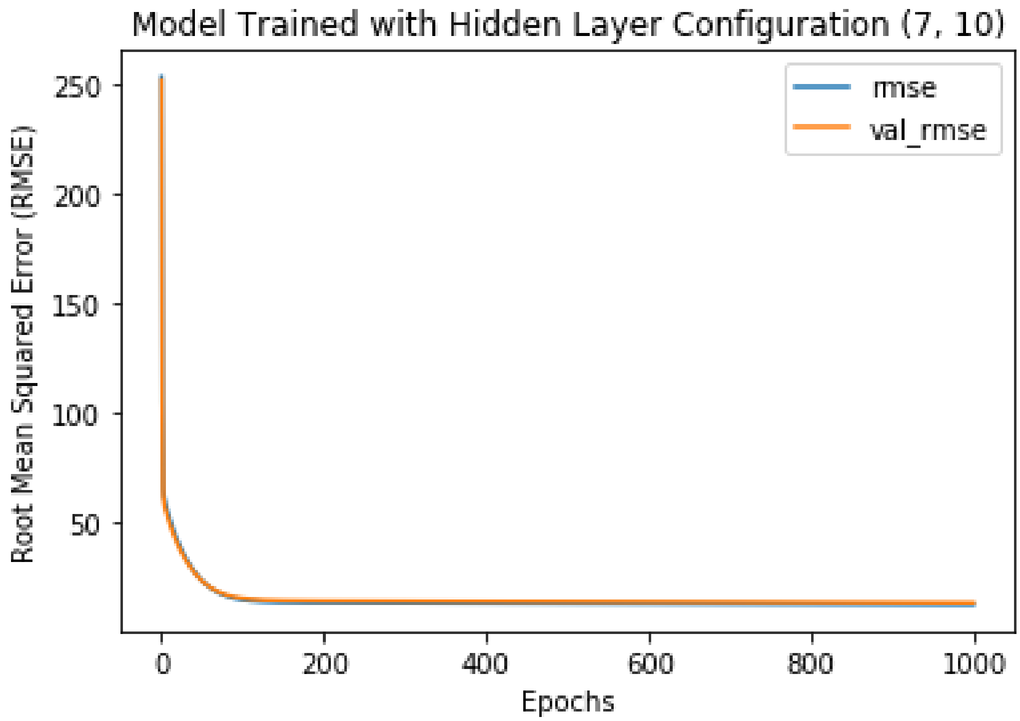
Model Trained with Hidden Layer Configuration (2, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
100
80
60
40
20
0
200
400
600
800
1000
Epochs

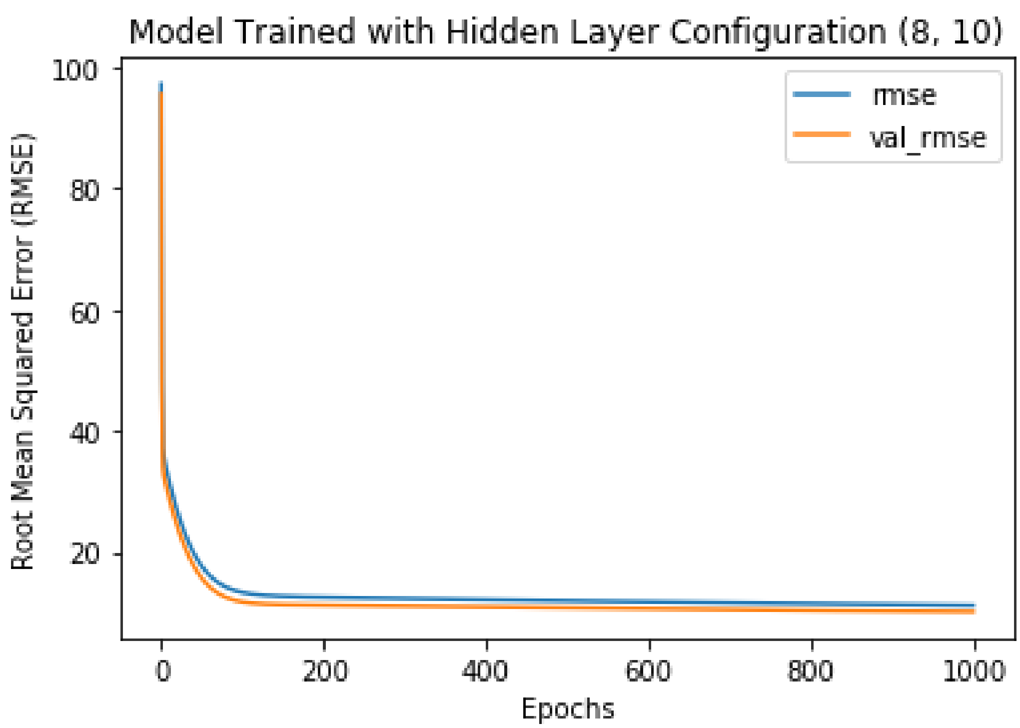

Model Trained with Hidden Layer Configuration (3, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
0
100
200
300
400
500
0
200
400
600
800
1000

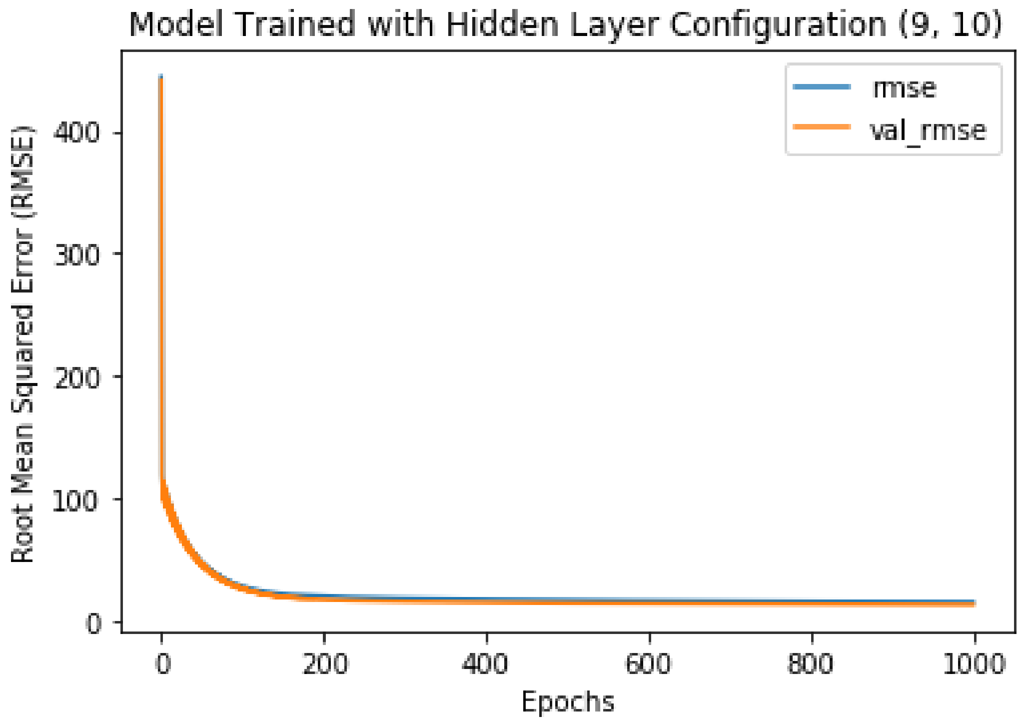

Model Trained with Hidden Layer Configuration (4, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
Epochs
50
100
150
200
0
200
400
600
800
1000

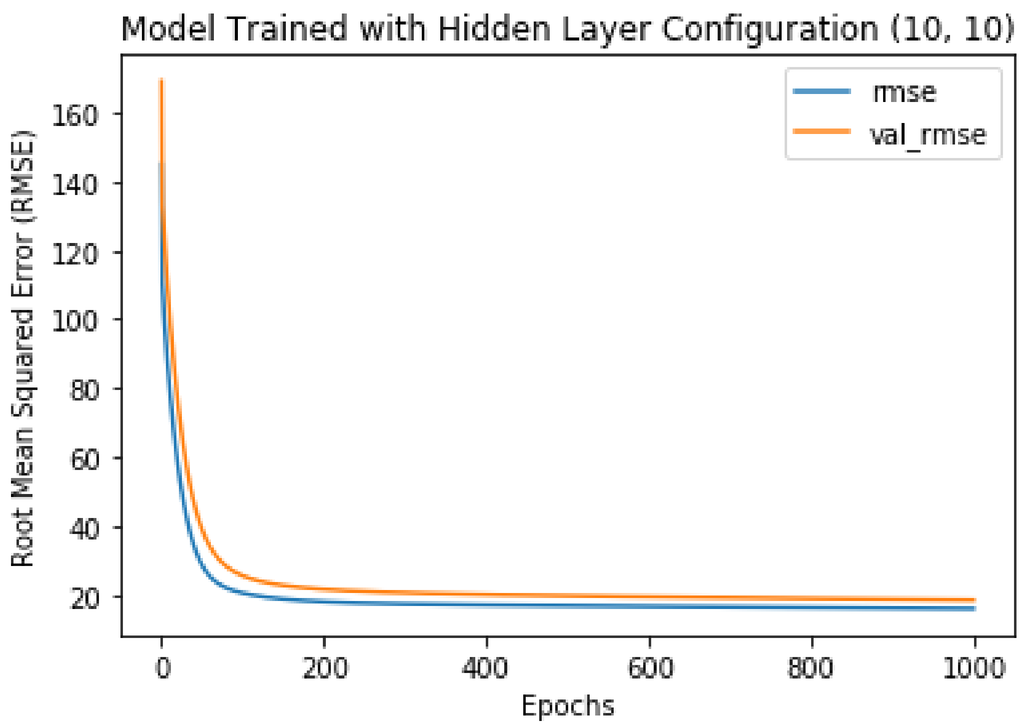

Model Trained with Hidden Layer Configuration (5, 10)
Root Mean Squared Error (RMSE)
Epochs
rmse
val_rmse
200
175
150
125
100
75
50
25
0
200
400
600
800
1000


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


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


Model Trained with Hidden Layer Configuration (8, 10)
Root Mean Squared Error (RMSE)
100
80
60
40
20
0
200
400
600
800
1000
Epochs
rmse
val_rmse


Model Trained with Hidden Layer Configuration (9, 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 (10, 10)
rmse
val_rmse
Root Mean Squared Error (RMSE)
160
140
120
100
80
60
40
20
0
200
400
600
800
1000
Epochs