3 Models are Built to check the accuracy:-

Model 1:- Base Model

```
[ ]  import numpy
     import pandas
     from keras.models import Sequential
     from keras.layers import Dense
     from keras.wrappers.scikit_learn import KerasRegressor
     from sklearn.model_selection import cross_val_score
     from sklearn.model_selection import KFold
     from sklearn.preprocessing import StandardScaler
     from sklearn.pipeline import Pipeline
```

```
[ ]  X = df
     Y = df_y
```

```
[ ]  # define base model
     def baseline_model():
       # create model
       model = Sequential()
       model.add(Dense(19, input_dim=19, kernel_initializer='normal', activation='relu'))
       model.add(Dense(1, kernel_initializer='normal'))
       # Compile model
       model.compile(loss='mean_squared_error', optimizer='adam')
       return model
```

```
[ ]  # fix random seed for reproducibility
     seed = 7
     numpy.random.seed(seed)
     # evaluate model with standardized dataset
     estimator = KerasRegressor(build_fn=baseline_model, epochs=100, batch_size=5, verbose=0)
```

```
[ ]  kfold = KFold(n_splits=10, random_state=seed)
     results = cross_val_score(estimator, X, Y, cv=kfold)
     print("Results: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

Result/Output:

```
Instructions for updating:

Use tf.cast instead.
```

`Results: -7.88 (17.24) MSE`

Model 2:- Standardised Values

```
# evaluate model with standardized dataset
numpy.random.seed(seed)
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasRegressor(build_fn=baseline_model, epochs=50, batch_size=5, verbose=0)))
pipeline = Pipeline(estimators)
kfold = KFold(n_splits=10, random_state=seed)
results = cross_val_score(pipeline, X, Y, cv=kfold)
print("Standardized: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

```
Standardized: -0.20 (0.11) MSE
```

Model 3: Large Model

```
[ ]  def larger_model():
       # create model
       model = Sequential()
       model.add(Dense(19, input_dim=19, kernel_initializer='normal', activation='relu'))
       model.add(Dense(9, kernel_initializer='normal', activation='relu'))
       model.add(Dense(1, kernel_initializer='normal'))
       # Compile model
       model.compile(loss='mean_squared_error', optimizer='adam')
       return model
```

```
[ ]  numpy.random.seed(seed)
     estimators = []
     estimators.append(('standardize', StandardScaler()))
     estimators.append(('mlp', KerasRegressor(build_fn=larger_model, epochs=50, batch_size=5, verbose=0)))
     pipeline = Pipeline(estimators)
     kfold = KFold(n_splits=10, random_state=seed)
     results = cross_val_score(pipeline, X, Y, cv=kfold)
     print("Larger: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

```
Larger: -2.48 (3.49) MSE
```

Using R:

```
> df=data.frame(WQI,Min_Val,Max_Val,Mean_Val,sd_Val)
> df
      WQI Min_Val  Max_Val     Mean_Val       sd_Val
1      pH    7.62     8.46     8.039168 2.420017e-01
2      TN    0.24     7.06     3.629908 1.980685e+00
3   BOD5     0.80     7.80     4.331888 2.018846e+00
4      TP    0.01     0.99     0.502216 2.811645e-01
5    NH3+    0.01     9.21     4.633614 2.653745e+00
6     COD    0.74   173.99    87.685750 5.032289e+01
7    Iron    0.01     1.65     0.845074 4.742745e-01
8  Copper    0.01     1.98     0.994540 5.706646e-01
9    Zinc    0.01     3.31     1.652162 9.486380e-01
10     DO    1.40    10.40     5.807422 2.627209e+00
11    TDS   90.04  9468.78  4759.639056 2.697320e+03
12     Ca   42.80  1082.08   567.597300 2.959269e+02
13     Mg    9.16  3766.38  1904.642600 1.085859e+03
14     Na    2.65  6749.43  3349.550998 1.935608e+03
15    Cl-   26.64  8837.79  4465.403206 2.577014e+03
16    HCO  100.91 24312.56 12186.282738 7.029578e+03
17    SO4    4.95  8420.46  4207.787900 2.404408e+03
18    PO4    0.00     1.70     0.855158 4.844554e-01
19     Cr    0.01     0.16     0.085112 4.401517e-02
>
```