**Data set 1 : Best Model:** Stacking-Best ML models-PCA

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **RMSE** | **MAE** | **MAD** | **MAPE** | **Best Params** | **Tuning Method** |
| Linear regression | 19.08 | 13.25 | 13.25 | 0.74 | intercept:10.996740896195206  Estiamted coefficients:[ 0.9850998 0.02252641 -0.01464863 -0.04430781 0.08095848 -0.1051743  0.00414234 0.12912902 -0.04883499 -0.03073708 -0.00730028 -0.00232372  0.05973975 0.00213758 -0.08925779 0.02714007 -0.02362059 0.07145879  -0.11268611 0.09064128 0.04023607 -0.0621883 -0.01158585 0.03744979  0.02689578 -0.07926611 0.02742816 0.11446739 -0.12280206 0.02950703] |  |
| Linear regression-Optimized | 19.08 | 13.25 | 13.25 | 0.74 | Best Parameters: {'fit\_intercept': True}  Coefficients: [ 0.9850998 0.02252641 -0.01464863 -0.04430781 0.08095848 -0.1051743  0.00414234 0.12912902 -0.04883499 -0.03073708 -0.00730028 -0.00232372  0.05973975 0.00213758 -0.08925779 0.02714007 -0.02362059 0.07145879  -0.11268611 0.09064128 0.04023607 -0.0621883 -0.01158585 0.03744979  0.02689578 -0.07926611 0.02742816 0.11446739 -0.12280206 0.02950703]  Intercept: 10.996740896195206 | Random Search 'fit\_intercept': [True, False] |
| Polynomial Regression -PCA | 47.71 | 37.02 | 29.72 | 2.04 % | Best Polynomial degree: 5 |  |
| Polynomial Regression-Optimized- PCA | 19.79 | 13.81 | 9.75 | 0.77% | Best parameters found: {'pca\_\_n\_components': 12, 'poly\_\_degree': 1} | Random Search 'poly\_\_degree': randint(1, 6) |
| Gaussian process | 42.014 | 29.113 | 19.645 | 1.613% | kernel = RBF(length\_scale=1.0) |  |
| Gaussian process-Optimized-PCA | 23.17 | 16.49 | 12.97 | 0.92 | {'kernel\_\_k1\_\_constant\_value': 3.845401188473625, 'kernel\_\_k2\_\_length\_scale': 9.60714306409916} | Random Search  'kernel\_\_k1\_\_constant\_value': uniform(0.1, 10.0),      'kernel\_\_k2\_\_length\_scale': uniform(0.1, 10.0) |
| MA-Uni | 201.73 | 133.53 | 133.53 | 7.650318316736248 % | window\_size = 5 |  |
| MA-Uni-Optimized | 117.870 | 89.832 | 89.832 | 5.172% | Best Window Size: 2 | Random Search  window\_sizes = range(2, 21) |
| MA-Multi | 159.71 | 125.47 | 125.47 | 7.21% | window\_size = 5 |  |
| MA-Multi-Optimized | 118.49 | 90.48 | 90.48 | 5.19% | Best Window Size: 2 | Random Search  window\_sizes = range(2, 21) |
| Arima-Uni | 187.20 | 139.40 | 139.40 | 8.36 | ARIMA(5, 1, 0) |  |
| Arima-Uni-Optimized | 175.198 | 135.245 | 135.245 | 7.833% | Best ARIMA(4, 2, 5) | Random Search  p\_range = range(0, 6)  d\_range = range(0, 3)  q\_range = range(0, 6) |
| Arima-Multi | 19.43 | 14.05 | 14.05 | 0.79% | SARIMAX(4,2,5) |  |
| Arima-Multi-Optimized | 18.949 | 13.116 | 13.116 | 0.737% | Best SARIMAX(4, 1, 1) | Random Search  p\_range = range(0, 6)  d\_range = range(0, 3)  q\_range = range(0, 6) |
| Stacking- With best STmodels | 19.17 | 13.33 | 8.92 | R-squared: 0.988 | ('gp', gp\_model),  ('poly', poly\_model),  ('sarimax', sarimax\_model),  ('lr', lr\_model) | # Create Gaussian Process Model  kernel = C(3.845, (1e-2, 1e2)) \* RBF(9.607, (1e-2, 1e2))  gp\_model = GaussianProcessRegressor(kernel=kernel, n\_restarts\_optimizer=9)  # Initialize models  sarimax\_model = CustomSARIMAX(order=(4, 1, 1))  poly\_model = CustomPolynomialRegression(degree=2)  lr\_model = LinearRegression() |
| Stacking- With best ST models-PCA | 18.92 | 13.20 | 9.06 | R-squared: 0.98 | ('gp', gp\_model),  ('poly', poly\_model),  ('sarimax', sarimax\_model),  ('lr', lr\_model) | # Create Gaussian Process Model  kernel = C(3.845, (1e-2, 1e2)) \* RBF(9.607, (1e-2, 1e2))  gp\_model = GaussianProcessRegressor(kernel=kernel, n\_restarts\_optimizer=9)  # Initialize models  sarimax\_model = CustomSARIMAX(order=(4, 1, 1))  poly\_model = CustomPolynomialRegression(degree=2)  lr\_model = LinearRegression() |
| KNN | 20.81 | 13.62 | 8.88 | 0.76 | Number of Neighbors: 3 | n\_neighbors = [3, 5, 7, 9, 11, 13, 15, 17] |
| KNN-Optimized | 19.72 | 12.66 | 8.75 | None% | {'n\_neighbors': 2, 'p': 2, 'weights': 'distance'} | Random Search  param\_distributions = {      'n\_neighbors': randint(1, 31),      'weights': ['uniform', 'distance'],      'p': [1, 2]  } |
| Decision Tree | 22.07 | 16.02 | 12.16 | 0.90 | Max Depth: 5 | max\_depths = [3, 5, 7, 9, 11, 13, 15, 17] |
| Decision Tree-Optimized | 21.92 | 15.82 | 12.15 | 0.88% | {'max\_depth': 5, 'min\_samples\_leaf': 5, 'min\_samples\_split': 8} | Random Search  param\_dist = {      'max\_depth': randint(3, 20),      'min\_samples\_split': randint(2, 20),      'min\_samples\_leaf': randint(1, 20)  } |
| SVR | 21.09 | 14.43 | 10.29 | 0.81% |  | kernel='rbf', C=1.0, epsilon=0.1 |
| SVR-Optimized | 19.02 | 12.94 | 8.62 | 0.72% | {'C': 88.62326508576255, 'epsilon': 0.020736445177905044, 'gamma': 0.00039459088111000007} | Random Search  param\_dist = {      'C': loguniform(1e-2, 1e2),      'gamma': loguniform(1e-4, 1e-1),      'epsilon': loguniform(1e-4, 1e-1)  } |
| Stacking-Best ML models | 18.8963 | 12.3529 | 8.4944 | -0.6970% | estimators=[  ('knn', knn),  ('decision\_tree', decision\_tree),  ('svr', svr)  ] | # Decision Tree with optimized parameters  decision\_tree = DecisionTreeRegressor(max\_depth=6, min\_samples\_leaf=14, min\_samples\_split=17)  # KNN with optimized parameters  knn = KNeighborsRegressor(n\_neighbors=2, p=2, weights='distance')  # SVR with optimized parameters  svr = make\_pipeline(StandardScaler(), SVR(C=88.623, epsilon=0.0207, gamma=0.000394)) |
| Stacking-Best ML models-PCA | 18.4848 | 12.1051 | 8.1582 | -0.6809% | estimators=[  ('knn', knn),  ('decision\_tree', decision\_tree),  ('svr', svr)  ] | n\_components = 15  # Number of PCA components to keep  decision\_tree = make\_pipeline(PCA(n\_components=n\_components),                                DecisionTreeRegressor(max\_depth=6, min\_samples\_leaf=14, min\_samples\_split=17))  knn = make\_pipeline(StandardScaler(), PCA(n\_components=n\_components),                      KNeighborsRegressor(n\_neighbors=2, p=2, weights='distance'))  svr = make\_pipeline(StandardScaler(), PCA(n\_components=n\_components),                      SVR(C=88.623, epsilon=0.0207, gamma=0.000394)) |
| GRU | 175.26 | 133.10 | 133.64 | 7.80% | Lag : 24  Dropout(rate=0.1)  First GRU layer: 200 units  Second GRU layer: 100 units  Third GRU layer: 50 units  Fourth GRU layer: 50 units |  |
| GRU-Optimized | 180.24 | 146.94 | 147.02 | 8.34% | {'model\_\_gru\_units4': 50, 'model\_\_gru\_units3': 75, 'model\_\_gru\_units2': 100, 'model\_\_gru\_units1': 300, 'model\_\_dropout\_rate': 0.4} | Random Search      'model\_\_gru\_units1': [100, 200, 300],      'model\_\_gru\_units2': [50, 100, 150],      'model\_\_gru\_units3': [25, 50, 75],      'model\_\_gru\_units4': [25, 50],      'model\_\_dropout\_rate': [0.1, 0.2, 0.3, 0.4, 0.5] |
| LSTM | 177.19 | 132.64 | 132.67 | 7.83% | Lag : 24  Dropout(rate=0.1)  First LSTM layer: 200 units  Second LSTM layer: 100 units  Third LSTM layer: 50 units  Fourth (last) LSTM layer: 50 units |  |
| LSTM-Optimized | 175.82 | 137.10 | 137.17 | 7.91% | {'model\_\_lstm\_units4': 25, 'model\_\_lstm\_units3': 25, 'model\_\_lstm\_units2': 50, 'model\_\_lstm\_units1': 100, 'model\_\_dropout\_rate': 0.2} | Random Search      'model\_\_lstm\_units1': [100, 200],      'model\_\_lstm\_units2': [50, 100],      'model\_\_lstm\_units3': [25, 50, 75],      'model\_\_lstm\_units4': [25, 50],      'model\_\_dropout\_rate': [0.1, 0.2] |
| BILSTM | 174.90 | 134.64 | 135.86 | 7.88% | Lag : 24  Dropout(rate=0.1)  First BILSTM layer: 200 units  Second BILSTM layer: 100 units  Third BILSTM layer: 50 units  Fourth (last) BILSTM layer: 50 units |  |
| BILSTM-Optimized | 177.03 | 140.71 | 141.16 | 8.08% | {'model\_\_dropout\_rate': 0.2, 'model\_\_bilstm\_units4': 25, 'model\_\_bilstm\_units3': 25, 'model\_\_bilstm\_units2': 100, 'model\_\_bilstm\_units1': 200} | Random Search      'model\_\_bilstm\_units1': [100, 200],      'model\_\_bilstm\_units2': [50, 100],      'model\_\_bilstm\_units3': [25, 50, 75],      'model\_\_bilstm\_units4': [25, 50],      'model\_\_dropout\_rate': [0.1, 0.2] |
| BIGRU | 178.18 | 144.01 | 144.55 | 8.23% | Lag : 24  Dropout(rate=0.1)  First BIGRU layer: 200 units  Second BIGRU layer: 100 units  Third BIGRU layer: 50 units  Fourth (last) BIGRU layer: 50 units |  |
| BIGRU-Optimized | 174.59 | 135.52 | 136.53 | 7.90% | {'model\_\_dropout\_rate': 0.2, 'model\_\_bigru\_units4': 50, 'model\_\_bigru\_units3': 50, 'model\_\_bigru\_units2': 100, 'model\_\_bigru\_units1': 100} | Random Search      'model\_\_bigru\_units1': [100, 200],      'model\_\_bigru\_units2': [50, 100],      'model\_\_bigru\_units3': [25, 50, 75],      'model\_\_bigru\_units4': [25, 50],      'model\_\_dropout\_rate': [0.1, 0.2] |
| CNN | 191.54 | 152.20 | 159.06 | 9.16% | Lag : 24  Filters1=64, kernel\_size=3, activation='relu'  Dropout(0.1)  Filters2=128, kernel\_size=3, activation='relu'  Dropout(0.1)  Filters3=128, kernel\_size=3, activation='relu'  MaxPooling1D(pool\_size=2) |  |
| CNN-Optimized | 189.40 | 154.40 | 166.25 | 9.38% | {'model\_\_kernel\_size2': 3, 'model\_\_kernel\_size1': 5, 'model\_\_filters2': 128, 'model\_\_filters1': 128, 'model\_\_dropout\_rate': 0.1} | Random Search      'model\_\_filters1': [32, 64, 128],      'model\_\_filters2': [64, 128, 256],      'model\_\_kernel\_size1': [3, 5],      'model\_\_kernel\_size2': [3, 5],      'model\_\_dropout\_rate': [0.1, 0.2, 0.3] |
| AutoEncoder with Best deep learning Model: BIGRU | 173.65 | 135.03 | 135.03 | 7.82 |  |  |

**Question 1:** I ran linear regression, Polynomial Regression with PCA (I used to reduce dimensions and improve performance), Moving average for Y and Moving Average with all variables, Arima for and Arima with all variables, and Gaussian process with PCA. In addition, in the plot of ACF and PACF, there is no significant seasonality.

**Question 2:** I run KNN(n\_neighbors = [3, 5, 7, 9, 11, 13, 15, 17]), Decision tree (max\_depths = [3, 5, 7, 9, 11, 13, 15, 17]) and SVR .

**Question 3:** I run GRU, LSTM, BILSTM, BIGRU, and CNN with lag 24.

**Question 4:** To tune models, I used a random search optimization. Random Search is used to efficiently explore a broad hyperparameter space for various deep learning models like GRU, LSTM, BiLSTM, BiGRU, and CNN and other machine learning models, offering a good balance between computational cost and parameter optimization.

In addition, to improve the best model of deep learning I used Bayesian Optimization.Bayesian Optimization then refines these models by targeting promising hyperparameter regions, enhancing model performance with fewer evaluations and higher precision. (All parameters are mentioned in the table)

**Question 5:**

* In some models I used PCA to reduce dimensions and improve them. In some models, I used stacking, stacking with PCA, AutoEncoder, and optimization Bayesian.
* Statistical model: I run Stacking- With best ST models including Linear regression, Polynomial Regression, Gaussian process with PCA, and Arima with best parameters, and Stacking- With best ST models - PCA. And, in the end, Stacking- With the best ST models-PCA is the best.
* Machine learning: I run Stacking-Best ML models and Stacking-Best ML models-PCA. And the best model is Stacking-Best ML models -PCA. (KNN, Decision tree, and SVR which are optimized with random search)

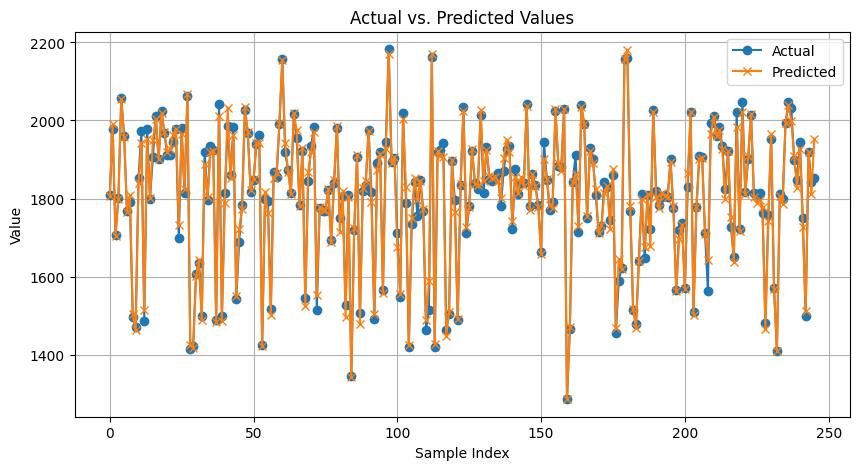
Deep learning: I run AutoEncoder with Best deep learning Model-BIGRU. And it is improved.

1. PCA (Principal Component Analysis): Used to reduce the dimensionality of the dataset, improving model training times and helping to prevent overfitting by removing less informative variables. PCA simplifies the complexity in high-dimensional data while retaining trends and patterns.
2. Stacking: This ensemble method combines multiple classification or regression models via a meta-classifier or meta-regressor. Stacking is used to increase the prediction accuracy by taking into account diverse perspectives from various models, leading to better generalization on unseen data.
3. Stacking with PCA: Integrating PCA into stacking reduces the input feature space, which can lead to more efficient learning and less overfitting in the stacked models. It makes the stacking ensemble not only faster but potentially more accurate by focusing on the most significant features.
4. AutoEncoder: Primarily used for unsupervised learning of efficient codings, AutoEncoders are helpful in dimensionality reduction and feature learning, especially useful in preparing data for complex predictive models by highlighting intrinsic structures or patterns in the data.
5. Bayesian Optimization: Employed for optimizing hyperparameters, Bayesian optimization uses a probabilistic model to guide the search for the best hyperparameters. This technique is particularly effective because it balances the exploration of the parameter space with the exploitation of known good regions, making it superior for finding optimal solutions efficiently.

**Question 6:**

* Statistical model: I run linear regression, Polynomial Regression with PCA (I used to reduce dimensions and improve performance), Moving average for Y and Moving Average with all variables, Arima for and Arima with all variables, and Gaussian process with PCA. After random search optimization, the best Model is Arima Multivariate (4, 1, 1).

Because, Metrices including RMSE, MAE, MAD, and MAPE are 18.949, 13.116, 13.116, 0.737%respectively, and the lowest.



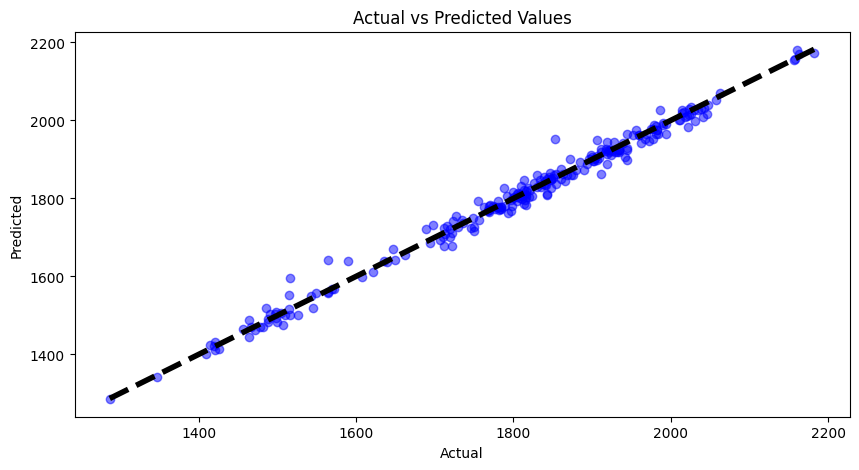
To improve statistical models, I run Stacking- With best ST models including Linear regression, Polynomial Regression, Gaussian process with PCA, and Arima with best parameters, and Stacking- With best ST models - PCA. Stacking- With best ST models - PCA is the best. The best model has metrics:

RMSE: 18.92

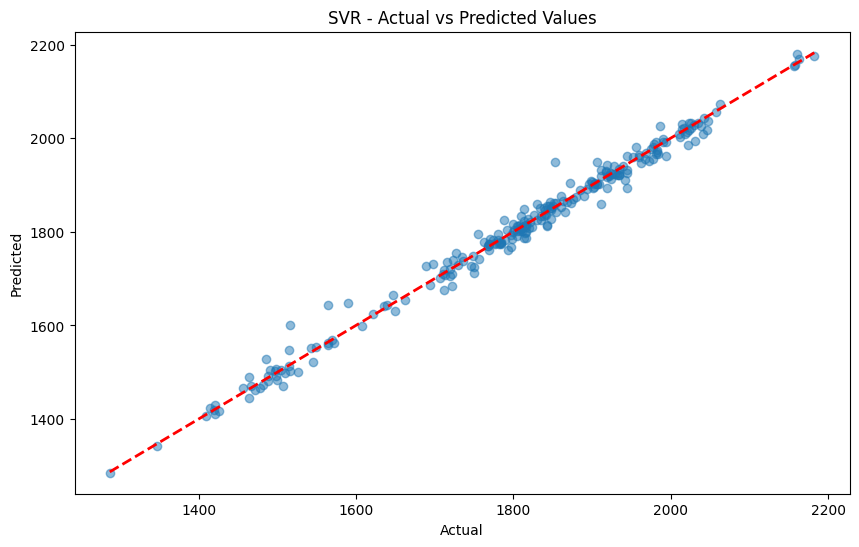
MAE: 13.20

MAD: 9.06

And the lowest.



* Machine learning: I run KNN(n\_neighbors = [3, 5, 7, 9, 11, 13, 15, 17]), Decision tree (max\_depths = [3, 5, 7, 9, 11, 13, 15, 17]) and SVR with PCA. After random search optimization, the best model is SVR -Optimized with {'C': 88.62326508576255, 'epsilon': 0.020736445177905044, 'gamma': 0.00039459088111000007}with metrics RMSE, MAE, MAD, MAPE: 19.02, 12.94, 8.62, 0.72% respectively, which is the lowest.



Then, to improve it, Stacking-Best ML models (KNN, Decision tree, and SVR which are optimized with random search), Stacking-Best ML models-PCA. Stacking-Best ML models-PCA is the best model. The metrics of the best model:

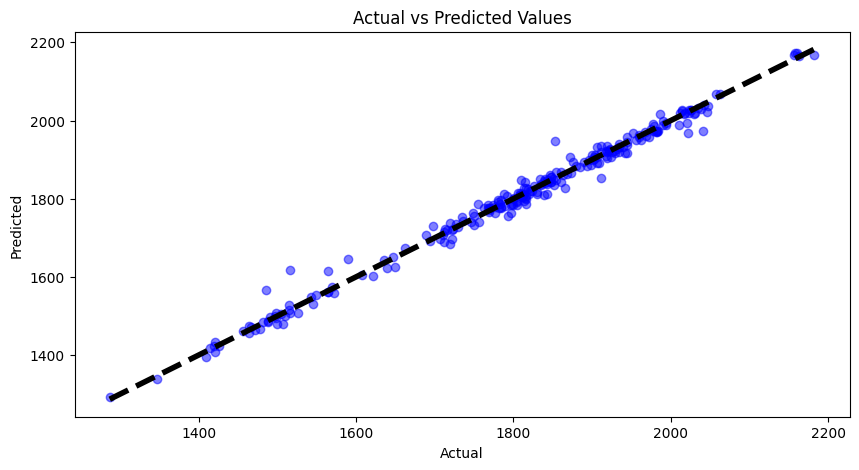
RMSE: 18.4848

MAE: 12.1051

MAD: 8.1582

MAPE : -0.6809%

Which is the lowest.



* Deep learning: I run GRU, LSTM, BILSTM, BIGRU, and CNN with lag 24. After random search optimization, the best model BIGRU-Optimized has RMSE, MAE, MAD, MAPE : 174.59,135.52,136.53, 7.90% respectively which is the lowest. Then to improve the best model, I tried AutoEncoder with the Best deep learning Model. And it is the best model.

RMSE: 173.65

MAE: 135.03

MAD: 135.03

MAPE: 7.82

All Models: the best model is The Stacking-Best ML models-PCA is the best model (KNN, Decision tree and SVR which are optimized with random search) with performance:

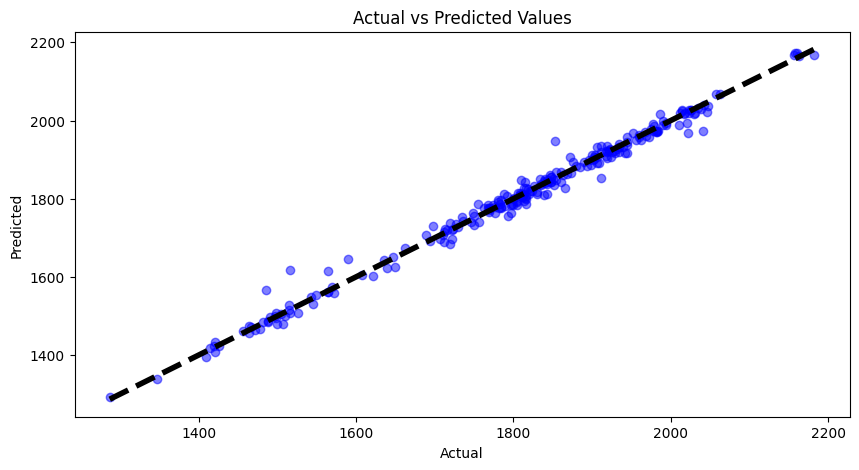
RMSE: 18.4848

MAE: 12.1051

MAD: 8.1582

MAPE : -0.6809%

Which is the lowest.



**Question 7:**

* Statistical model: After random search, the worst models including Arima and Moving Average (except Arima-Multi) are not good because they have the highest metrics.
* Machine learning: After random search optimization, Decision Tree{'max\_depth': 5, 'min\_samples\_leaf': 5, 'min\_samples\_split': 8}has the highest metrics including RMSE, MAE, MAD, MAPE: 21.92,15.82, 12.15, 0.88% respectively.

Deep learning: After random search process, the worst model is CNN {'model\_\_kernel\_size2': 3, 'model\_\_kernel\_size1': 5, 'model\_\_filters2': 128, 'model\_\_filters1': 128, 'model\_\_dropout\_rate': 0.1} with the highest metrics including RMSE, MAE, MAD, and MAPE: 189.40,154.40, 166.25, 9.38% respectively.

All Models: the worst model is CNN with the highest metrics including RMSE, MAE, MAD, and MAPE: 189.40,154.40, 166.25, 9.38% respectively.