GHGs Emissions data future predictions using machine learning

Introduction

Here I discuss my machine learning capstone project I applied various algorithms (for both classification and regression) to identify the most reliable algorithm that depicts the highest performance on both training and test data and that can be considered for the future dataset.

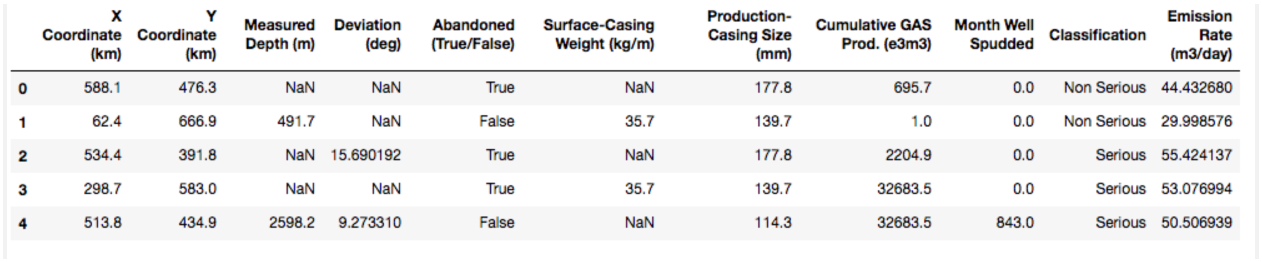
Dataset

The data file "*GHG_Emission.csv*" has been retrieved from AER website; where the locations of the wells have been changed, and some key properties are generated synthetically or are greatly manipulated for confidential reasons.

Regression and Classification

Gathering Data

First, the dataset was imported and read using pandas. The data was shuffled and then random. seed (42) was used to save the state of a random function. The index of the data was reset.



	X Coordinate (km)	Y Coordinate (km)	Measured Depth (m)	Deviation (deg)	Abandoned (True/False)	Surface-Casing Weight (kg/m)	Production-Casing Size (mm)	Cumulative GAS Prod. (e3m3)	Month Well Spudded	Classification	Emission Rate (m3/day)
0	588.1	476.3	NaN	NaN	True	NaN	177.8	695.7	0.0	Non Serious	44.432680
1	62.4	666.9	491.7	NaN	False	35.7	139.7	1.0	0.0	Non Serious	29.998576
2	534.4	391.8	NaN	15.690192	True	NaN	177.8	2204.9	0.0	Serious	55.424137
3	298.7	583.0	NaN	NaN	True	35.7	139.7	32683.5	0.0	Serious	53.076994
4	513.8	434.9	2598.2	9.273310	False	NaN	114.3	32683.5	843.0	Serious	50.506939

Data Processing

Stratified sampling was performed for even distribution of data. The test and training data were split based on that.

The outliers were removed for instances out of the range of $\mu \pm 2.5\sigma$, imputation (with median) was performed, text handling using one-hot encoding and standardization.

CLASSIFICATION

Model Training for Classification

Binary classification was applied using the following Machine Learning models below.

- Dummy Classifier
- Stochastic Gradient Descent

- Logistic Regression
- Support Vector Machine: Linear
- Support Vector Machine: Polynomial Kernel
- Decision Trees
- Random Forest
- Adaptive Boosting with Linear SVM
- Adaptive Boosting
- Hard and Soft Voting
- Shallow Neural Network (with 3 layers)
- Deep Neural Network (with 6 layers)

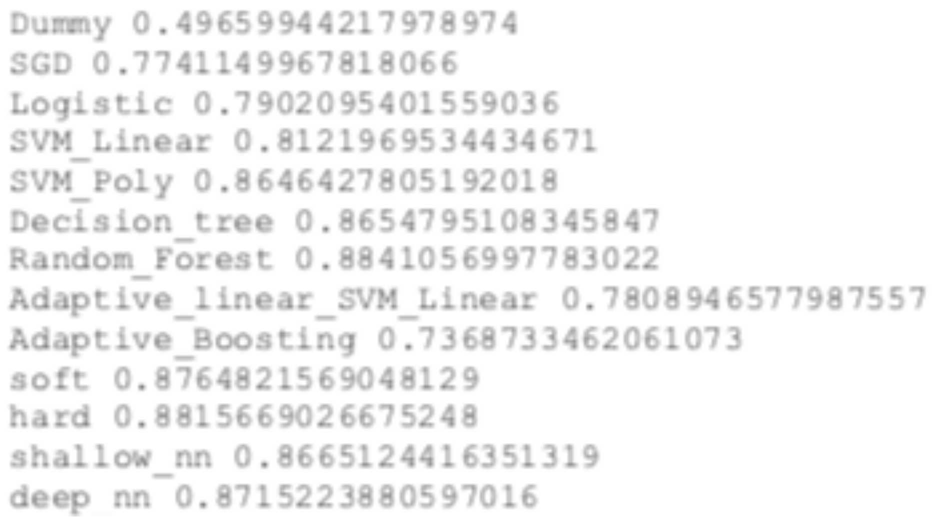
The hyperparameters were fine-tuned using *RandomizedSearchCV* based on *accuracy*. The optimized parameters were used to predict accuracy. K-fold cross-validation with 5-folds (cv=5) was applied and then the mean of 5 accuracies for each classifier was calculated.

These optimized hyper-parameters for all the above-mentioned algorithms were used for finding the performance on the test dataset as well.

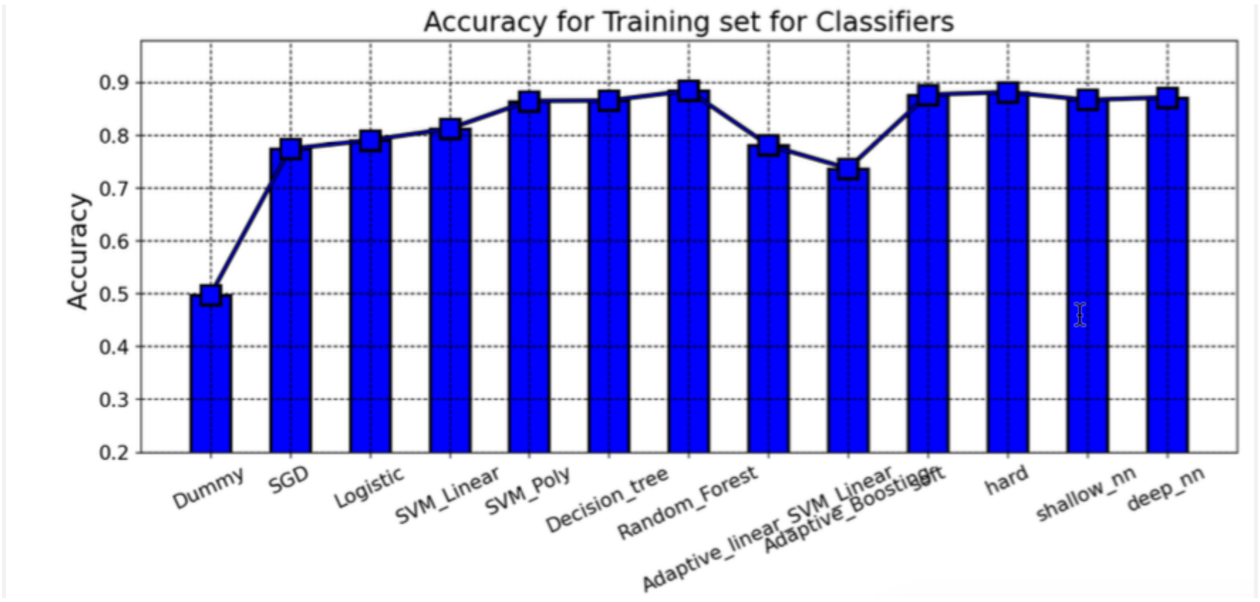
Model Performance for Regression

TRAINING DATASET

Here are the summary table and graph depicting the performance on the training dataset.



```
Dummy 0.49659944217978974
SGD 0.7741149967818066
Logistic 0.7902095401559036
SVM_Linear 0.8121969534434671
SVM_Poly 0.8646427805192018
Decision_tree 0.8654795108345847
Random_Forest 0.8841056997783022
Adaptive_linear_SVM_Linear 0.7808946577987557
Adaptive_Boosting 0.7368733462061073
soft 0.8764821569048129
hard 0.8815669026675248
shallow_nn 0.8665124416351319
deep_nn 0.8715223880597016
```

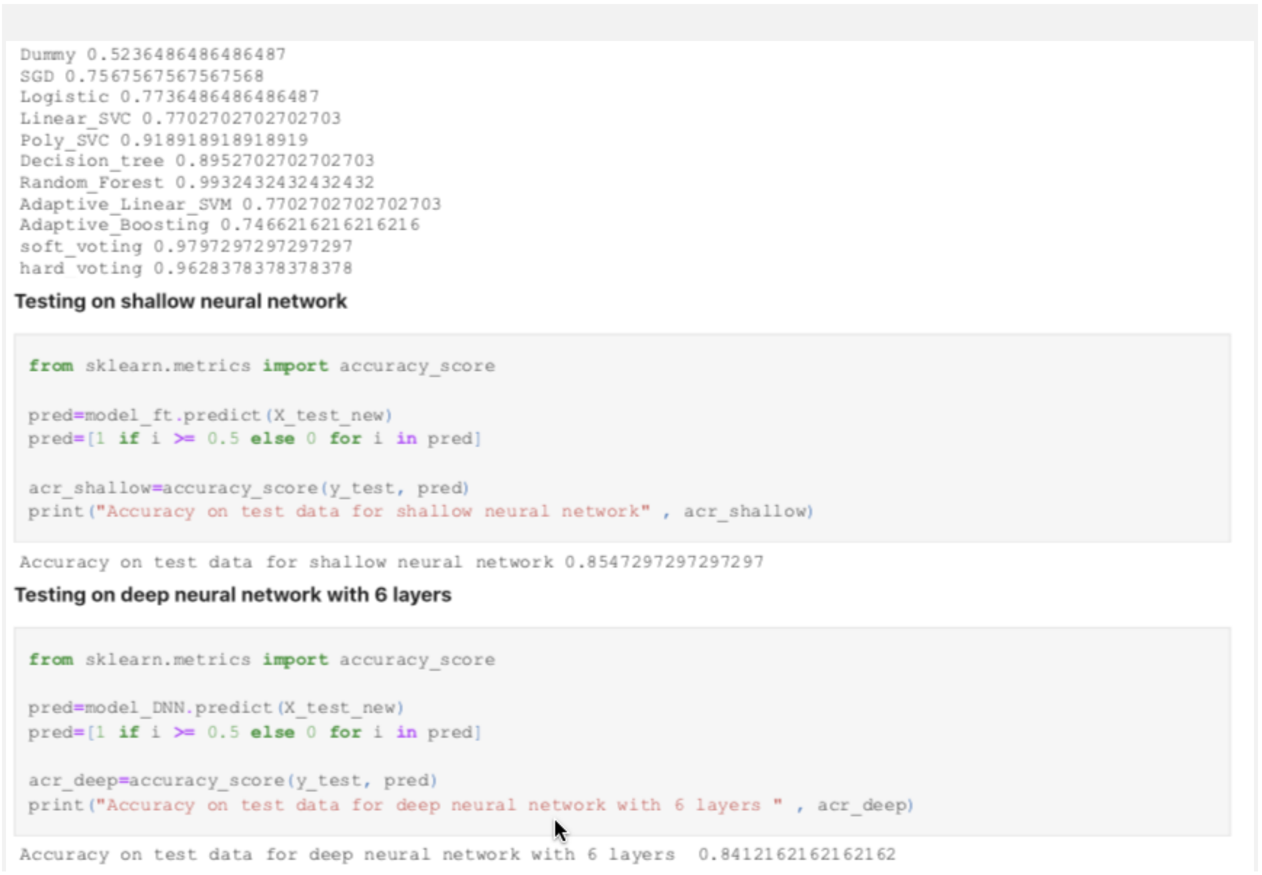


.

Clearly, Random Forest is the best algorithm on the training dataset.

TESTING DATASET

Here are the summary table and graph depicting the performance on the training dataset.



```
Dummy 0.5236486486486487
SGD 0.7567567567567568
Logistic 0.7736486486486487
Linear_SVC 0.7702702702702703
Poly_SVC 0.918918918918919
Decision_tree 0.8952702702702703
Random_Forest 0.9932432432432432
Adaptive_Linear_SVM 0.7702702702702703
Adaptive_Boosting 0.7466216216216216
soft_voting 0.9797297297297297
hard_voting 0.9628378378378378
```

Testing on shallow neural network

```
from sklearn.metrics import accuracy_score

pred=model_ft.predict(X_test_new)
pred=[1 if i >= 0.5 else 0 for i in pred]

acr_shallow=accuracy_score(y_test, pred)
print("Accuracy on test data for shallow neural network" , acr_shallow)
```

```
Accuracy on test data for shallow neural network 0.8547297297297297
```

Testing on deep neural network with 6 layers

```
from sklearn.metrics import accuracy_score

pred=model_DNN.predict(X_test_new)
pred=[1 if i >= 0.5 else 0 for i in pred]

acr_deep=accuracy_score(y_test, pred)
print("Accuracy on test data for deep neural network with 6 layers " , acr_deep)
```

```
Accuracy on test data for deep neural network with 6 layers  0.8412162162162162
```

Clearly, Random Forest is the best algorithm on the testing dataset.

Conclusion for Classification

Random Forest should be used for future datasets as it gives the best performance on both testing and training data.

REGRESSION

Model Training for Regression

Similar to Binary classification, Regression was applied with the following Machine Learning models below.

- Linear Regression
- Support Vector Machine: Polynomial Kernel
- Decision Trees
- Random Forest
- Gradient Boosting
- Shallow Neural Network (with 3 layers)

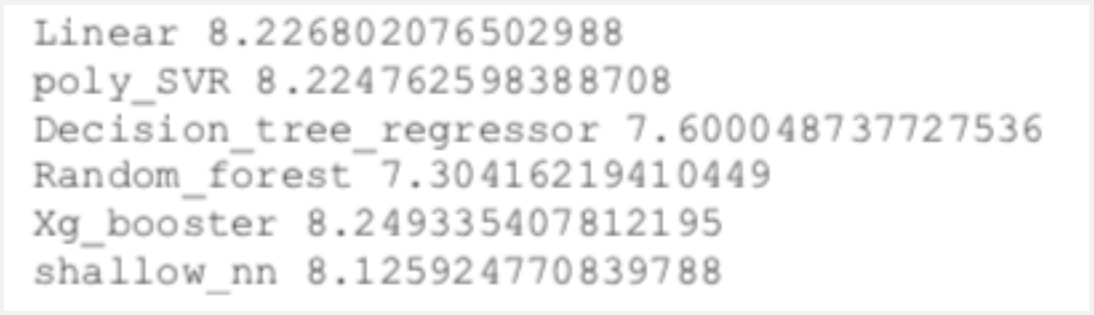
The hyperparameters were fine-tuned using *RandomizedSearchCV* based on *RMSE.* The optimized parameters were used to predict RMSE. K-fold cross-validation with 5-folds (cv=5) was applied and then the mean of 5 RMSEs for each regressor was calculated.

These optimized hyper-parameters for all the above-mentioned algorithms were used for finding the performance on the test dataset as well.

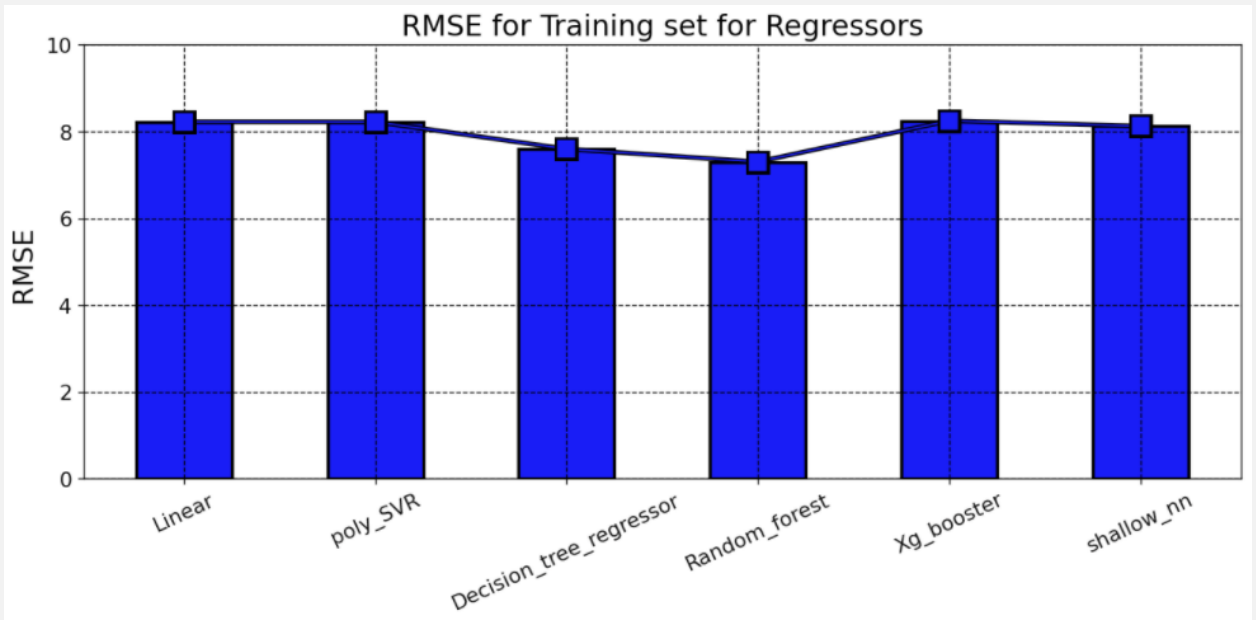
Model Performance for Regression

TRAINING DATASET

Here are the summary table and graph depicting the performance on the training dataset.



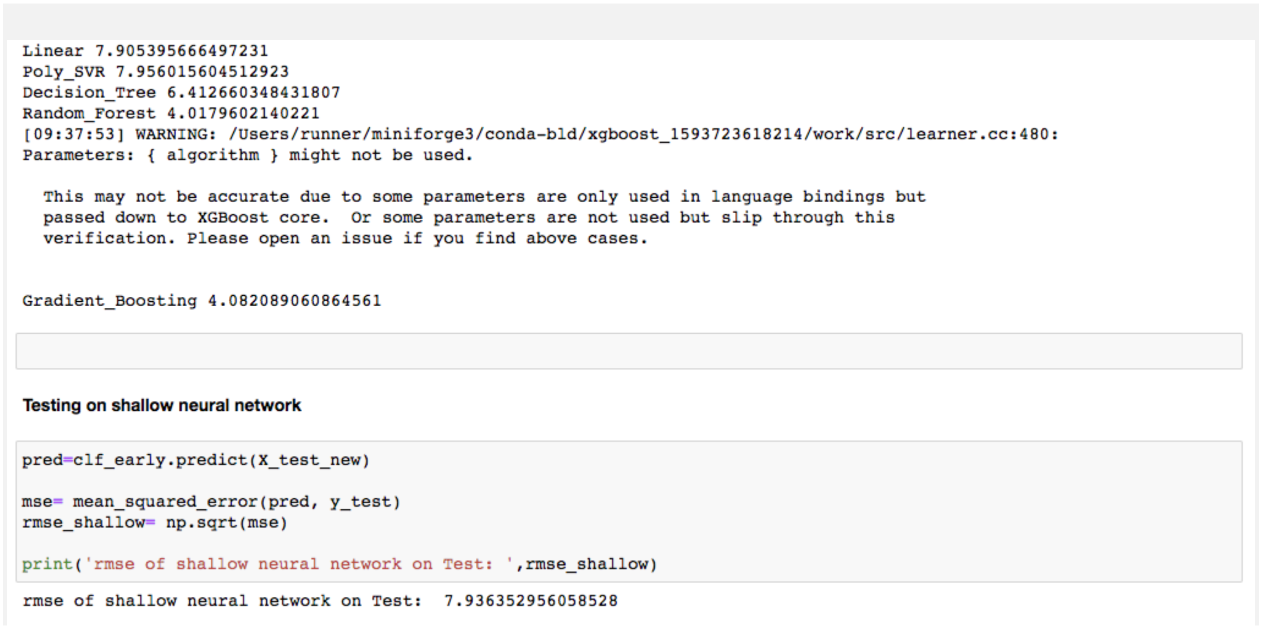
```
Linear 8.226802076502988
poly_SVR 8.224762598388708
Decision_tree_regressor 7.600048737727536
Random_forest 7.30416219410449
Xg_booster 8.249335407812195
shallow_nn 8.125924770839788
```



Clearly, Random Forest is the best algorithm on the training dataset.

TESTING DATASET

Here are the summary table and graph depicting the performance on the testing dataset.



```
Linear 7.905395666497231
Poly_SVR 7.956015604512923
Decision_Tree 6.412660348431807
Random_Forest 4.0179602140221
[09:37:53] WARNING: /Users/runner/miniforge3/conda-bld/xgboost_1593723618214/work/src/learner.cc:480:
Parameters: { algorithm } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


Gradient_Boosting 4.082089060864561
```

Testing on shallow neural network

```
pred=clf_early.predict(X_test_new)

mse= mean_squared_error(pred, y_test)
rmse_shallow= np.sqrt(mse)

print('rmse of shallow neural network on Test: ',rmse_shallow)
```

```
rmse of shallow neural network on Test:  7.936352956058528
```

Performance on the testing dataset

Clearly, Random Forest is the best algorithm on the testing dataset.

Conclusion for Regression

Random Forest should be used for future datasets as it gives the best performance on both testing and training data.