# ML- To be sent

November 24, 2024

[7]:
```
from sklearn import preprocessing
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
from seaborn import boxplot
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestRegressor
#from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import train_test_split
import time
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.decomposition import PCA
import xgboost as xgb
from sklearn.linear_model import LinearRegression
```

[5]:
```
%pip install xgboost
```

```
Collecting xgboostNote: you may need to restart the kernel to use updated
packages.

  Downloading xgboost-2.1.2-py3-none-win_amd64.whl.metadata (2.1 kB)
Requirement already satisfied: numpy in h:\anaconda\lib\site-packages (from
xgboost) (1.26.4)
Requirement already satisfied: scipy in h:\anaconda\lib\site-packages (from
xgboost) (1.13.1)
Downloading xgboost-2.1.2-py3-none-win_amd64.whl (124.9 MB)
   ---------------------------------------- 0.0/124.9 MB ? eta -:--:--
   - -------------------------------------- 3.9/124.9 MB 19.5 MB/s eta 0:00:07
   -- ------------------------------------- 8.7/124.9 MB 20.6 MB/s eta 0:00:06
   ---- ----------------------------------- 13.4/124.9 MB 21.0 MB/s eta 0:00:06
   ----- ---------------------------------- 18.1/124.9 MB 21.5 MB/s eta 0:00:05
   ------- -------------------------------- 22.5/124.9 MB 21.3 MB/s eta 0:00:05
```

```
   -------- ------------------------------ 26.5/124.9 MB 21.0 MB/s eta 0:00:05
   --------- ------------------------------ 31.2/124.9 MB 21.1 MB/s eta 0:00:05
   ---------- ----------------------------- 35.9/124.9 MB 21.3 MB/s eta 0:00:05
   ----------- ---------------------------- 40.4/124.9 MB 21.4 MB/s eta 0:00:04
   ------------- -------------------------- 45.4/124.9 MB 21.5 MB/s eta 0:00:04
   --------------- ------------------------ 50.1/124.9 MB 21.5 MB/s eta 0:00:04
   ----------------- ---------------------- 54.5/124.9 MB 21.6 MB/s eta 0:00:04
   ------------------ --------------------- 59.2/124.9 MB 21.7 MB/s eta 0:00:04
   -------------------- ------------------- 63.7/124.9 MB 21.7 MB/s eta 0:00:03
   --------------------- ------------------ 67.4/124.9 MB 21.7 MB/s eta 0:00:03
   ----------------------- ---------------- 73.1/124.9 MB 21.7 MB/s eta 0:00:03
   ------------------------ --------------- 77.3/124.9 MB 21.7 MB/s eta 0:00:03
   -------------------------- ------------- 81.8/124.9 MB 21.6 MB/s eta 0:00:02
   --------------------------- ------------ 86.0/124.9 MB 21.7 MB/s eta 0:00:02
   ---------------------------- ----------- 90.4/124.9 MB 21.7 MB/s eta 0:00:02
   ------------------------------ --------- 94.9/124.9 MB 21.7 MB/s eta 0:00:02
   ------------------------------- -------- 99.1/124.9 MB 21.7 MB/s eta 0:00:02
   -------------------------------- ------- 103.5/124.9 MB 21.7 MB/s eta 0:00:01
   --------------------------------- ------ 108.3/124.9 MB 21.7 MB/s eta 0:00:01
   ---------------------------------- ----- 112.2/124.9 MB 21.6 MB/s eta 0:00:01
   ----------------------------------- ---- 116.7/124.9 MB 21.6 MB/s eta 0:00:01
   ----------------------------------- ---- 121.6/124.9 MB 21.7 MB/s eta 0:00:01
   ------------------------------------  124.8/124.9 MB 21.7 MB/s eta 0:00:01
   ------------------------------------- 124.9/124.9 MB 21.2 MB/s eta 0:00:00
Installing collected packages: xgboost
Successfully installed xgboost-2.1.2
```

[9]:
```
data=pd.read_csv('Data/Iowa1.csv')
data.head()
```

[9]:

| | year | yday | dayl (s) | prcp (mm/day) | srad (W/m^2) | swe (kg/m^2) | \ |
|---|---|---|---|---|---|---|---|
| 0 | 1982 | 1 | 32486.40039 | 4 | 185.600006 | 24 | |
| 1 | 1982 | 2 | 32486.40039 | 9 | 176.000000 | 32 | |
| 2 | 1982 | 3 | 32486.40039 | 10 | 169.600006 | 40 | |
| 3 | 1982 | 4 | 32486.40039 | 9 | 124.800003 | 52 | |
| 4 | 1982 | 5 | 32486.40039 | 0 | 252.800003 | 52 | |

| | tmax (deg c) | tmin (deg c) | vp (Pa) | Soil type | … | \ |
|---|---|---|---|---|---|---|
| 0 | -7.5 | -22.0 | 120 | Ely silty clay loam | … | |
| 1 | -1.0 | -14.0 | 200 | Ely silty clay loam | … | |
| 2 | -2.0 | -14.0 | 200 | Ely silty clay loam | … | |
| 3 | -7.0 | -14.5 | 200 | Ely silty clay loam | … | |
| 4 | -2.0 | -16.5 | 160 | Ely silty clay loam | … | |

| | Sand Content % | Clay Content % | Silt Content % | \ |
|---|---|---|---|---|
| 0 | 5 | 28.3 | 66.7 | |
| 1 | 5 | 28.3 | 66.7 | |

```
2                5            28.3            66.7
3                5            28.3            66.7
4                5            28.3            66.7

    soil bulk density (grams per cubic centimeter)   wilting point %  \
0                                             1.33              17.7
1                                             1.33              17.7
2                                             1.33              17.7
3                                             1.33              17.7
4                                             1.33              17.7

   field capacity %  saturation point (cm)  Yield (27.6gN/m2)  \
0              31.5                   122                0.0
1              31.5                   122                0.0
2              31.5                   122                0.0
3              31.5                   122                0.0
4              31.5                   122                0.0

   Yield (11.6gN/m2)  Yield (40.1gN/m2)
0                  0                0.0
1                  0                0.0
2                  0                0.0
3                  0                0.0
4                  0                0.0

[5 rows x 22 columns]
```

[11]:
```python
#Data drop for US data
data = data.drop('year',axis=1)
data = data.drop('Soil type',axis=1)
data = data.drop('swe (kg/m^2)',axis=1)
data = data.drop('Soil pH',axis=1)
data = data.drop('Yield (40.1gN/m2)',axis=1)
data = data.drop('Yield (11.6gN/m2)',axis=1)
data = data.drop(' soil bulk density (grams per cubic centimeter)',axis=1)
data = data.drop(' wilting point %',axis=1)
data = data.drop('field capacity %',axis=1)
data = data.drop('saturation point (cm)',axis=1)
data = data.drop('Soil Organic matter %',axis=1)
data = data.drop('Sand Content %',axis=1)
data = data.drop('Clay Content %',axis=1)
data = data.drop('Silt Content %',axis=1)

#Data drop for X.0 data
#data = data.drop('Soil_pH',axis=1)
#data = data.drop('Date/Time',axis=1)
#data = data.drop('Yield (6gN/m2)',axis=1)
```

```
#data = data.drop('Yield (27.6gN/m2)',axis=1)
#data = data.drop('Yield (40.1gN/m2)',axis=1)
#data = data.drop('Sand Content %',axis=1)
#data = data.drop('Clay Content %',axis=1)
#data = data.drop('Silt Content %',axis=1)



data.head()
```

[11]:

| | yday | dayl (s) | prcp (mm/day) | srad (W/m^2) | tmax (deg c) | tmin (deg c) | \ |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 32486.40039 | 4 | 185.600006 | -7.5 | -22.0 | |
| 1 | 2 | 32486.40039 | 9 | 176.000000 | -1.0 | -14.0 | |
| 2 | 3 | 32486.40039 | 10 | 169.600006 | -2.0 | -14.0 | |
| 3 | 4 | 32486.40039 | 9 | 124.800003 | -7.0 | -14.5 | |
| 4 | 5 | 32486.40039 | 0 | 252.800003 | -2.0 | -16.5 | |

| | vp (Pa) | Yield (27.6gN/m2) |
|---|---|---|
| 0 | 120 | 0.0 |
| 1 | 200 | 0.0 |
| 2 | 200 | 0.0 |
| 3 | 200 | 0.0 |
| 4 | 160 | 0.0 |

[13]:
```
x_scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
y_scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
x_data = data.values[:, :-1]
y_data = data.values[:, -1].reshape((-1, 1))
x_scaler.fit(x_data)
y_scaler.fit(y_data)
x_r_data = x_scaler.transform(x_data)
y_r_data = y_scaler.transform(y_data)
```

[15]:
```
X_, y = x_r_data[:], y_r_data.reshape((1, -1))[0]
```

[17]:
```
pca_model = PCA(n_components=3)
X = pca_model.fit_transform(X_)
```

[19]:
```
models = {}
#models["SVM"] = SVR(kernel ='rbf', max_iter=300, C=1, tol=0.001)
models["MLP"] = MLPRegressor(solver='adam', activation='relu', max_iter=1000,␣
 ↪learning_rate_init=0.001, hidden_layer_sizes=(1, 1, 1))
models["R. Forest"] = RandomForestRegressor(n_estimators=100)
##models["gradient"] = GradientBoostingRegressor()
#models["Xgboost"] = xgb.XGBRegressor(objective ='reg:squarederror',␣
 ↪colsample_bytree = 0.3, learning_rate = 0.8,max_depth = 1, alpha = 1,␣
 ↪n_estimators = 100)
```

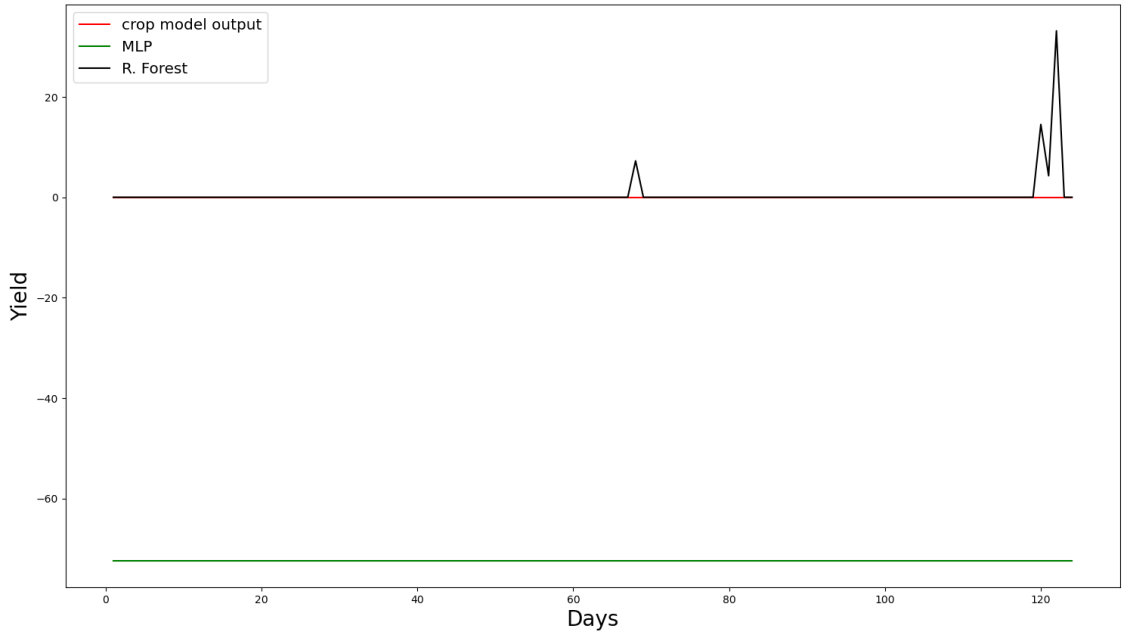
```
predictions = {}
colors = {"MLP":'green', "R. Forest":'black'} #,'Xgboost':'orange', "SVM":
 ↪'blue'} # "gradient":'brown'
par_mse = []
par_rmse = []
par_rrmse = []
par_mae = []
par_r2 = []
performance_names = ["MSE", "RMSE", "RRMSE", "MAE", "R2"]
model_names = []
```

[21]:
```
for name in models:
    model_names.append(name)
    models[name].fit(X,y)
    y_pred = cross_val_predict(models[name], X, y,cv=3)
    predictions[name] = models[name].predict(X)
    mse = mean_squared_error(y, y_pred)
    rmse =math.sqrt(mse)
    rrmse=rmse/np.mean(y)
    mae=mean_absolute_error(y, y_pred)
    r2=r2_score(y, y_pred)
    par_mse.append(mse)
    par_rmse.append(rmse)
    par_rrmse.append(rrmse)
    par_mae.append(mae)
    par_r2.append(r2)
    print('{:8s}'.format(name), ' =>  RMSE = {:.4f}'.format(rmse),' RRMSE = {:.
 ↪4f}'.format(rrmse) ,' MAE = {:.4f}'.format(mae), ' R2 = {:.4f}'.format(r2))␣
 ↪#MSE = {:.4f}'.format(mse)
```

```
MLP       =>  RMSE = 0.2595  RRMSE = 3.9563  MAE = 0.1659  R2 = -0.5657
R. Forest  =>  RMSE = 0.5739  RRMSE = 8.7497  MAE = 0.3910  R2 = -6.6583
```

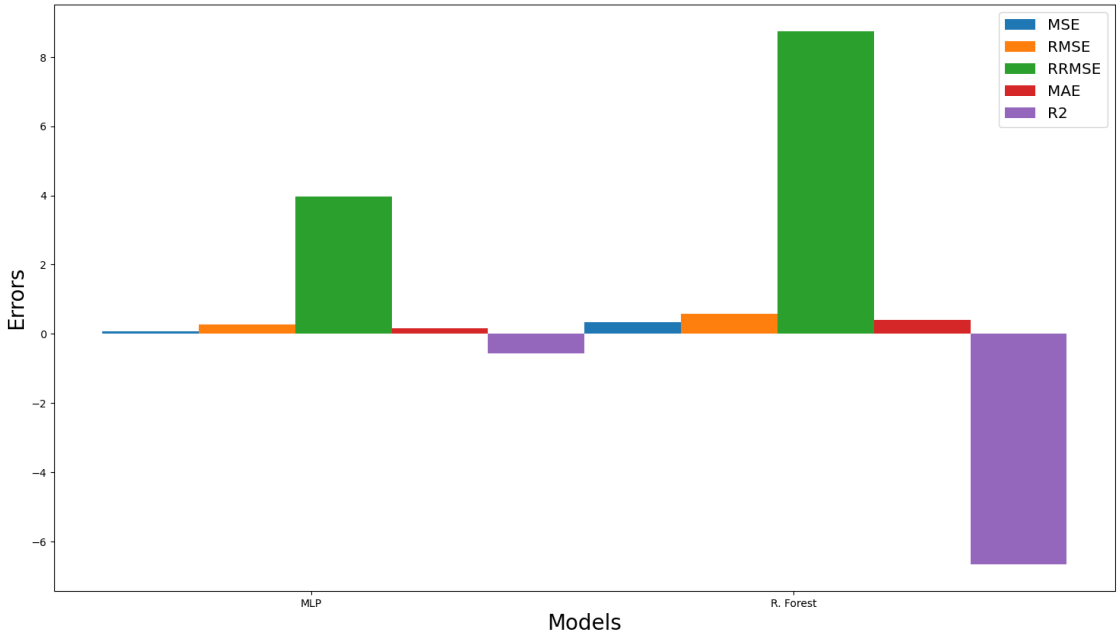
[23]:
```
t_range = [i for i in range(1, 125)]
plt.figure(figsize=(18, 10))
plt.plot(t_range, y_data[t_range], color='red', label="crop model output")
for name in predictions:
    inv_preds = y_scaler.inverse_transform(predictions[name].reshape((-1, 1))).
 ↪reshape((1, -1))[0]
    plt.plot(t_range, inv_preds[t_range], color=colors[name], label=name)
plt.legend(fontsize="x-large")
plt.xlabel('Days', fontsize=20)
plt.ylabel('Yield', fontsize=20)
plt.show
```

[23]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```




[25]:
```
X_axis = np.arange(len(model_names))
plt.figure(figsize=(18, 10))
width = 0.2
r_width = 0
par = [par_mse, par_rmse, par_rrmse,par_mae, par_r2]
for i, name in enumerate(performance_names):
    plt.bar(X_axis + r_width, par[i], width=width, label = name)
    r_width +=width

plt.xticks(X_axis+r_width/3, model_names)
plt.xlabel("Models", fontsize=20)
plt.ylabel("Errors", fontsize=20)
plt.legend(fontsize="x-large")
plt.show()
```


MSE
RMSE
RRMSE
MAE
R2
Errors
8
6
4
2
0
−2
−4
−6
MLP
R. Forest
Models

[ ]: