Predicting Corn, Wheat and Soybean Yield

Maura Tokay | UMBC DATA606

Exploratory Data Analyses

This dataset is part of the Farming System Project at USDA, Beltsville MD
https://www.ars.usda.gov/northeast-area/beltsville-md-barc/beltsville-agricultural-research-center/sustainable-agricultural-systems-laboratory/docs/farming-systems-project/

This data is not available online on the USDA website but can be found on my GitHub
https://github.com/mmtokay/DATA606/tree/master/datasets

Data is split in 2 files: crop and weather information

Exploratory Data Analyses

There is no crop data for 1999, this year Maryland had a drought and because the project did not use irrigation, crops never matured.

```
# Calculate duration between PlantingDate and HarvestDate
data['weekDuration'] = data['HarvestDate'] - data['PlantingDate']
data['weekDuration'] = data['weekDuration']/np.timedelta64(1,'W')
print('\nCheck unique values for Crop, GrowingSeason and SystemName columns.\n')
print("Crop", data.Crop.unique())
print("\nGrowing Season", data.GrowingSeason.unique())
print("\nCrop Management Type", data.SystemName.unique())
```

```
Check unique values for Crop, GrowingSeason and SystemName columns.

Crop ['CRN' 'SOY' 'WHT']

Growing Season [1996 1997 1998 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
 2011 2012 2013 2014 2015 2016]

Crop Management Type ['NT' 'CT' 'Org2' 'Org3' 'Org6' 'ORG2' 'ORG3' 'ORG6']
```

Exploratory Data Analyses

Duplicate crop management system.

```
# Calculate duration between PlantingDate and HarvestDate
data['weekDuration'] = data['HarvestDate'] - data['PlantingDate']
data['weekDuration'] = data['weekDuration']/np.timedelta64(1,'W')
print('\nCheck unique values for Crop, GrowingSeason and SystemName columns.\n')
print("Crop", data.Crop.unique())
print("\nGrowing Season", data.GrowingSeason.unique())
print("\nCrop Management Type", data.SystemName.unique())
```

```
Check unique values for Crop, GrowingSeason and SystemName columns.

Crop ['CRN' 'SOY' 'WHT']

Growing Season [1996 1997 1998 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
 2011 2012 2013 2014 2015 2016]

Crop Management Type ['NT' 'CT' 'Org2' 'Org3' 'Org6' 'ORG2' 'ORG3' 'ORG6']
```

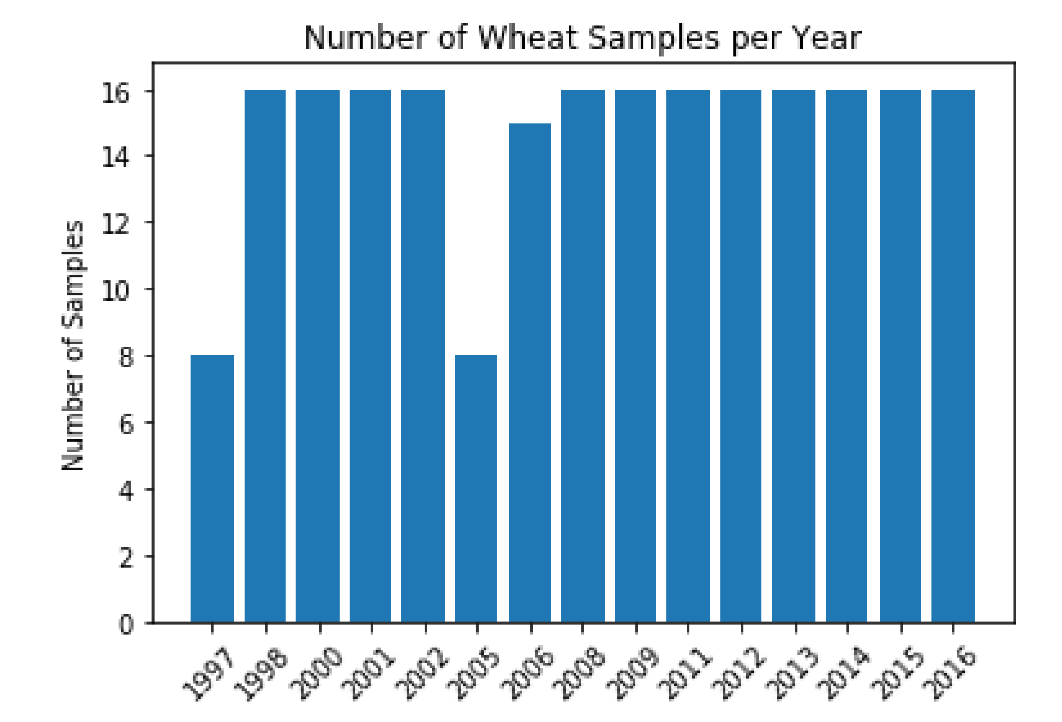
```
data['SystemName'] = data['SystemName'].str.upper()
print("\nCrop Management Type", data.SystemName.unique())
```

```
Crop Management Type ['NT' 'CT' 'ORG2' 'ORG3' 'ORG6']
```

```
# 1 for conventional
# 0 for organic
data['SystemNameType'] = ((data.SystemName == "NT") | (data.SystemName == "CT")).map({True:'1', False:'0'})
# Drop SystemName column
data.drop('SystemName', axis=1, inplace=True)
```

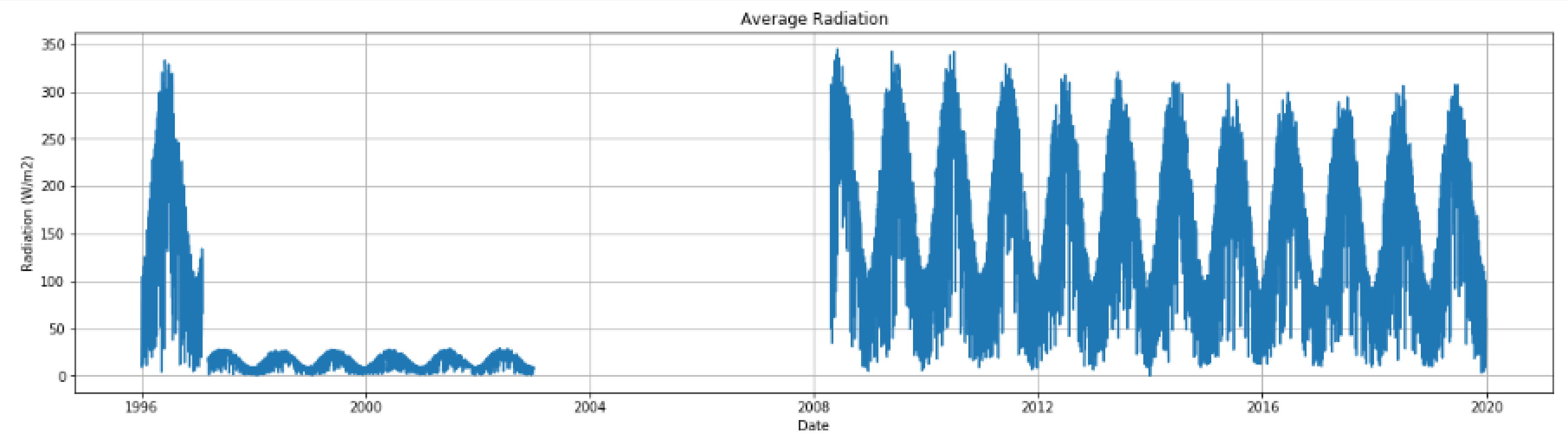
Exploratory Data Analyses

Wheat does not have crop data for 1996, 1999, 2003, 2004, 2007, 2010.



Exploratory Data Analyses

Average radiation will not be used because data is missing for years 2003-2008.



Feature Engineering

Week duration

$$\text{weekDuration} = \frac{(harvestDate - plantingDate)}{7}$$

```
# Calculate duration between PlantingDate and HarvestDate
data['weekDuration'] = data['HarvestDate'] - data['PlantingDate']
data['weekDuration'] = data['weekDuration']/np.timedelta64(1,'W')
```

Minimum week duration for corn, soybean and wheat are respectively, 16, 15 and 31 weeks.

Feature Engineering

Growing Degree Days (GDD)

$$\text{GDD} = \sum_{i=0}^{n}\left(\frac{T_{max} - T_{min}}{2}\right) - T_b$$

The base temperature for corn and soybean is 10°C and for wheat is 4.4°C.

```
def calcGDD(df,startDate,endDate,factor):
    gdd = 0
    for i, j in df.loc[(df.Date >= startDate) & (df.Date <= endDate)].iterrows():
        gdd = gdd + (((j['maxTempC']+j['minTempC'])/2)-factor)
    return gdd
```

Feature Engineering

Four functions were created to group daily weather data to weekly.

```
def calcMax(df,startDate,endDate,var):
    val = []
    for i, j in df.loc[(df.Date >= startDate) & (df.Date <= endDate)].iterrows():
        val.append(j[var])
    maxVal = max(val)
    return maxVal
```

```
def calcMin(df,startDate,endDate,var):
    val = []
    for i, j in df.loc[(df.Date >= startDate) & (df.Date <= endDate)].iterrows():
        val.append(j[var])
    minVal = min(val)
    return minVal
```

```
def calcAverage(df,startDate,endDate,var):
    sum = 0
    avg = 0
    for i, j in df.loc[(df.Date >= startDate) & (df.Date <= endDate)].iterrows():
        sum = sum + j[var]
    if sum > 0:
        avg = sum/(i+1)
    return avg
```

```
def calcSum(df,startDate,endDate,var):
    sum = 0
    for i, j in df.loc[(df.Date >= startDate) & (df.Date <= endDate)].iterrows():
        sum = sum + j[var]
    return sum
```

Feature Engineering

Create a matrix with weather features and the target data that it is yield.

```
def createFeaturesMatrix(cropData,weatherData,numWeeks,GDDFactor):
    master_tp = list()
    colName = ()
    i = 0
    j = 0
    for i, j in cropData.iterrows():
        if (i == 0):
            startDate = j['PlantingDate']
        #start calculating date ranges to aggregate weather data for 16 weeks starting from plantingDate
        new_tp = ()
        for w in range(numWeeks):
            temp_tuple = ()
            beginWeek = j['PlantingDate'] + timedelta(days=7)*w
            endWeek = j['PlantingDate'] + timedelta(days=7)*(w+1)
            if(w==(numWeeks-1)):
                temp_tuple = (calcAverage(weather_data,beginWeek,endWeek,'avgtTempC'),\
                              calcMax(weather_data,beginWeek,endWeek,'maxTempC'),\
                              calcMin(weather_data,beginWeek,endWeek,'minTempC'),\
                              calcMax(weather_data,beginWeek,endWeek,'maxHumPct'),\
                              calcMin(weather_data,beginWeek,endWeek,'minHumPct'),\
                              calcAverage(weather_data,beginWeek,endWeek,'meanWindMs-1'),\
                              calcSum(weather_data,beginWeek,endWeek,'PrecipitationMm'),\
                              calcGDD(weather_data,startDate,endWeek,GDDFactor),\
                              j['SystemNameType'],j['GrainYield'])
                if (i == 0):
                    colName = colName + ('avgTemp'+str(w+1),'maxTemp'+str(w+1),'minTemp'+str(w+1),\
                                         'maxHum'+str(w+1),'minHum'+str(w+1),'meanWind'+str(w+1),\
                                         'Precip'+str(w+1),'GDD','SystemNameType','GrainYield')
            else:
                temp_tuple = (calcAverage(weather_data,beginWeek,endWeek,'avgtTempC'),\
                              calcMax(weather_data,beginWeek,endWeek,'maxTempC'),\
                              calcMin(weather_data,beginWeek,endWeek,'minTempC'),\
                              calcMax(weather_data,beginWeek,endWeek,'maxHumPct'),\
                              calcMin(weather_data,beginWeek,endWeek,'minHumPct'),\
                              calcAverage(weather_data,beginWeek,endWeek,'meanWindMs-1'),\
                              calcSum(weather_data,beginWeek,endWeek,'PrecipitationMm'))
                if (i == 0):
                    colName = colName + ('avgTemp'+str(w+1),'maxTemp'+str(w+1),'minTemp'+str(w+1),\
                                         'maxHum'+str(w+1),'minHum'+str(w+1),'meanWind'+str(w+1),\
                                         'Precip'+str(w+1))
            new_tp = new_tp + temp_tuple
        #print(new_tp)
        master_tp.append(new_tp)

    new_df = pd.DataFrame(list(master_tp),columns = colName)
    return(new_df)
```

Data Files - Corn

A function was created to build the file with weather features and target data (crop yield).

```
new_df16 = createFeaturesMatrix(data_corn,weather_data,16,10)
new_df16.to_csv(r'cornFeatures16w.csv', index = False, header=True)
```

```
new_df15 = createFeaturesMatrix(data_corn,weather_data,15,10)
new_df15.to_csv(r'cornFeatures15w.csv', index = False, header=True)
```

```
new_df14 = createFeaturesMatrix(data_corn,weather_data,14,10)
new_df14.to_csv(r'cornFeatures14w.csv', index = False, header=True)
```

Corn (390 rows)

16 weeks
114 features + 1 target columns

15 weeks
107 features + 1 target columns

14 weeks
100 features + 1 target columns

Data Files - Soybean

A function was created to build the file with weather features and target data (crop yield).

```
new_soy_df15 = createFeaturesMatrix(data_soy,weather_data,15,10)
new_soy_df15.to_csv(r'soyFeatures15w.csv', index = False, header=True)
```

```
new_soy_df14 = createFeaturesMatrix(data_soy,weather_data,14,10)
new_soy_df14.to_csv(r'soyFeatures14w.csv', index = False, header=True)
```

```
new_soy_df13 = createFeaturesMatrix(data_soy,weather_data,13,10)
new_soy_df13.to_csv(r'soyFeatures13w.csv', index = False, header=True)
```

Soybean (500 rows)

15 weeks
107 features + 1 target columns

14 weeks
100 features + 1 target columns

13 weeks
93 features + 1 target columns

Data Files - Wheat

A function was created to build the file with weather features and target data (crop yield).

```
new_wheat_df31 = createFeaturesMatrix(data_wheat,weather_data,31,4.4)
new_wheat_df31.to_csv(r'wheatFeatures31w.csv', index = False, header=True)
```

```
new_wheat_df30 = createFeaturesMatrix(data_wheat,weather_data,30,4.4)
new_wheat_df30.to_csv(r'wheatFeatures30w.csv', index = False, header=True)
```

```
new_wheat_df29 = createFeaturesMatrix(data_wheat,weather_data,29,4.4)
new_wheat_df29.to_csv(r'wheatFeatures29w.csv', index = False, header=True)
```

Wheat (223 rows)

31 weeks
219 features + 1 target columns

30 weeks
212 features + 1 target columns

29 weeks
205 features + 1 target columns

Model Construction – Data Normalization

Used RobustScaler to better handle the outliers. The algorithm removes the median and scales the data using the 1st and 3rd quartile for each feature independently.

```
# Splitting data set
train_X, test_X = train_test_split(dataCorn16w.drop('GrainYield', axis=1), random_state=1)
train_y, test_y = train_test_split(dataCorn16w['GrainYield'], random_state=1)

# Apply Robust Scaler
scaler = RobustScaler()
train_scaler_X = scaler.fit_transform(train_X)
test_scaler_X = scaler.transform(test_X)
```

Regression Algorithm - Lasso

Lasso regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients.

```
lm = linear_model.Lasso(alpha=0.6)
lm.fit(train_scaler_X,train_y)

y_pred = lm.predict(test_scaler_X)

modelEvaluation(test_y, y_pred)

important_features = pd.Series(data=lm.coef_,index=dataCorn16w.drop('GrainYield', axis=1).columns)
important_features.sort_values(ascending=False,inplace=True)
print(important_features[:10])
print(important_features[-10:])
```

```
Mean absolute error regression loss (Best is 0) = 929.64779
Mean squared error (Best is 0) = 1388279.04679
Median absolute error regression loss (Best is 0) = 801.01523
Coefficient of determination (Best is 1) = 0.82528
SystemNameType    2332.068185
minHum11          1160.282462
minHum12           963.612618
minTemp11          903.056762
minHum7            785.077530
```

Regression Algorithms – Decision Tree Regressor

Decision tree regression builds a classification model in the form of a tree structure. The topmost decision node in a tree corresponds to the best predictor called root node.

```
tree_model = DecisionTreeRegressor()
tree_model.fit(train_scaler_X,train_y)

y_pred = tree_model.predict(test_scaler_X)

modelEvaluation(test_y, y_pred)

mportant_features = pd.Series(data=tree_model.feature_importances_,index=dataCorn16w.drop('GrainYield', axis=1).columns)
important_features.sort_values(ascending=False,inplace=True)
print(important_features[:10])
print(important_features[-10:])
```

```
Mean absolute error regression loss (Best is 0) = 930.40608
Mean squared error (Best is 0) = 1408752.93658
Median absolute error regression loss (Best is 0) = 715.18750
Coefficient of determination (Best is 1) = 0.82596
SystemNameType    2332.068185
minHum11          1160.282462
minHum12           963.612618
minTemp11          903.056762
```

Regression Algorithms – Random Forest Regressor

Random Forest Regressor is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

```
rf_model = RandomForestRegressor()
rf_model.fit(train_scaler_X,train_y)

y_pred = rf_model.predict(test_scaler_X)
modelEvaluation(test_y, y_pred)

important_features = pd.Series(data=rf_model.feature
important_features.sort_values(ascending=False,inpla
print(important_features[:10])
print(important_features[-10:])
```

```
Mean absolute error regression loss (Best is 0) = 97
Mean squared error (Best is 0) = 1515128.49119
Median absolute error regression loss (Best is 0) =
Coefficient of determination (Best is 1) = 0.80931
maxTemp6          0.306132
maxTemp9          0.108667
maxTemp8          0.100709
maxTemp10         0.057258
```

```
max_depth = 35
n_est = 500
rf_model = RandomForestRegressor(n_estimators=n_est,max_depth=max_depth,random_state=0)
rf_model.fit(train_scaler_X,train_y)

y_pred = rf_model.predict(test_scaler_X)
modelEvaluation(test_y, y_pred)

important_features = pd.Series(data=rf_model.feature_importances_,index=dataCorn16w.drop('GrainYield', axis=1).columns)
important_features.sort_values(ascending=False,inplace=True)
print(important_features[:10])
print(important_features[-10:])
```

```
Mean absolute error regression loss (Best is 0) = 928.11560
Mean squared error (Best is 0) = 1385379.13656
Median absolute error regression loss (Best is 0) = 772.60945
Coefficient of determination (Best is 1) = 0.82587
maxTemp9          0.234800
maxTemp6          0.212729
maxTemp8          0.049778
SystemNameType    0.042538
```

Preliminary Results - Corn

The best result for corn is Random Forest Regressor model for 15 weeks.

Weeks	Mean Absolute Error	Mean Squared Error	Median Absolute Error	Coefficient of Determination	Feature Importance
16	928.115	1385379.136	772.609	0.826	maxTemp9 0.23 maxTemp6 0.21 maxTemp8 0.05
15	925.964	1382744.199	772.609	0.826	maxTemp9 0.24 maxTemp6 0.21 maxTemp8 0.05
14	948.281	1448582.602	767.763	0.817	maxTemp9 0.40 maxHum13 0.11 maxTemp4 0.04

Table 1. *Corn Random Forest Regressor (customized) results for 16, 15 and 14 weeks of data.*

Preliminary Results - Soybean

The best result for soybean is for Random Forest Regressor model for 14 weeks.

Weeks	Mean Absolute Error	Mean Squared Error	Median Absolute Error	Coefficient of Determination	Feature Importance	
15	371.249	242497.114	270.451	0.814	minTemp12 maxTemp7 minTemp5	0.47 0.13 0.06
14	358.551	224661.315	284.613	0.835	minTemp12 maxTemp7 minTemp5	0.47 0.10 0.06
13	366.674	234401.733	252.278	0.826	minTemp12 minTemp5 maxTemp7	0.48 0.11 0.11

Table 2. *Soybean Random Forest Regressor results for 15, 14 and 13 weeks of data.*

Preliminary Results - Wheat

The best result for wheat is for Lasso model for 31 weeks.

Weeks	Mean Absolute Error	Mean Squared Error	Median Absolute Error	Coefficient of Determination	Feature Importance
31	437.308	315384.795	417.106	0.715	minHum7 313.14 minHum8 298.33 SystemNameType 281.49
30	437.323	315378.236	417.103	0.714	minHum7 317.22 minHum8 298.55 SystemNameType 281.51
29	437.352	315416.478	416.873	0.715	minHum7 323.74 minHum8 294.89 SystemNameType 281.50

***Table 3**. Wheat Lasso regression results for 31, 30 and 29 weeks of data.*

Next Steps

Do more work with Feature Engineering working with different number of weeks for each crop.

Drop features with the least weight to check if the this will impact model performance.

Acknowledgment

Michel Cavigelli, PhD, USDA

Anne E Conklin, USDA

More Information

https://github.com/mmtokay/DATA606

Questions?

Contact Maura Tokay at matokay1@umbc.edu