Machine Learning 101

Data Science Team

08/26/2020

Andrew Wheeler, PhD

andrew.wheeler@hms.com

Agenda

- Prediction vs Inference
- Supervised Learning Models
- Other Models
- Example Regression vs Random Forests in Python

Prediction Vs Inference

- Traditional Statistics mostly cares about obtaining *unbiased* parameter estimates to make inferences, e.g. estimating the β's in a regression equation:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

Definitions	
$\hat{y}$	Prediction from model
β	Regression coefficient
x	Independent variable

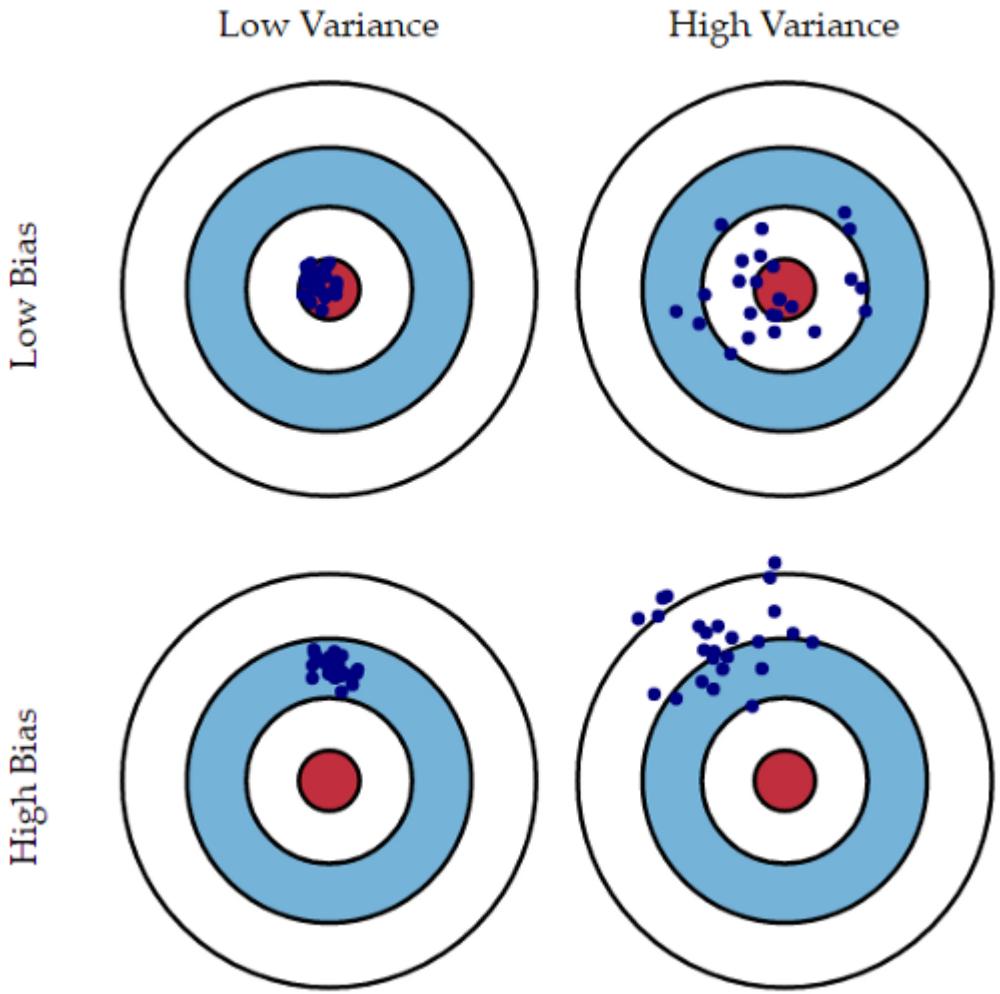
- Machine Learning mostly cares about obtaining *good* predictions, $\hat{y}$:

$$\hat{y} = f(x_1, x_2, x_3)$$

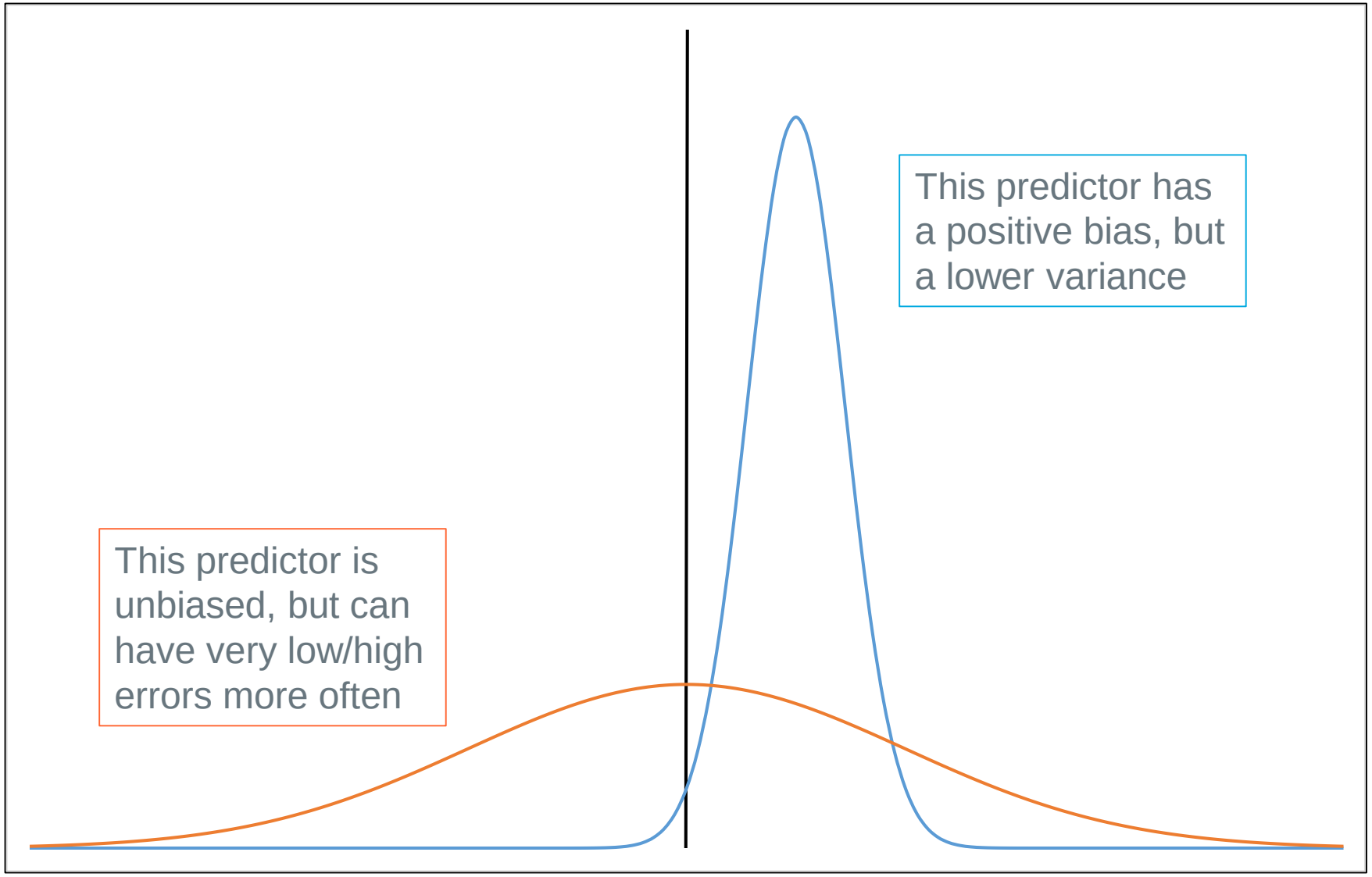
Bias vs Variance

- Sometimes to get *unbiased* estimates, you need to increase to the *variance* of the predictions. In practice, you may prefer a biased estimate with a lower variance.

- Machine Learning is mostly about finding algorithms to make good predictions, even if they are biased.



Bias Vs Variance



Cost Functions

- Cost (or Loss) functions are how you tell if your model is *good*.
- You try to minimize these via your model

- Examples:
  - Traditional linear regression loss function squared error, $\text{minimize}\{\Sigma\,(y_i - \hat{y}_i)^2\}$
  - L1 regression (absolute error), $\text{minimize}\{\Sigma|y_i - \hat{y}_i|\}$

- Can be specific to business needs, e.g.:
  - false negatives (e.g. loss of potential revenue) have a higher loss than false positives (e.g. person time to audit claim)
  - Over predictions are penalized more than under predictions (*The Price is Right Rules*)

Limitations of Linear Regression

- Regression can be used to make predictions:
  - Linear to predict a continuous outcome
  - Logistic to predict categorical outcomes (yes/no, or multiple categories)

- Regression has some strong limitations in practice:
  - Need to know the functional form of the model upfront

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_3 + \beta_5 (x_2 \cdot x_3)$$

  - Here specified x_1 has a quadratic effect, and x_2 and x_3 have an interaction
  - In practice you don't know what the form *should be*, and testing all possibilities is very difficult (e.g. x_1 may be a cubic function, not a quadratic. There may be a three way interactions between all $x's$, etc.)

Limitations of Machine Learning

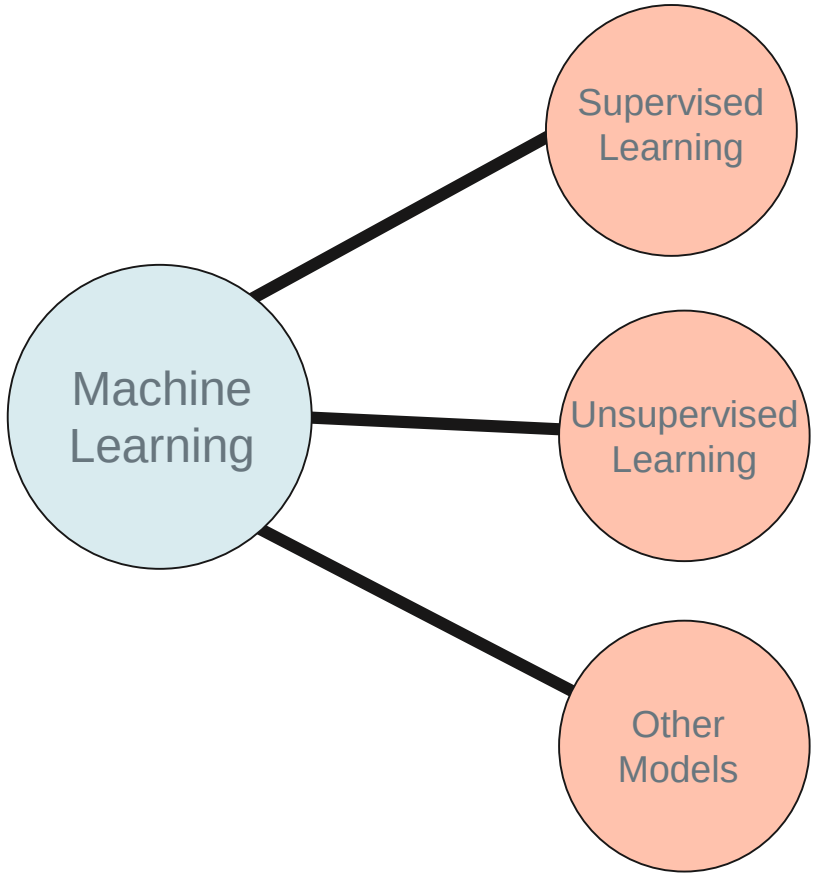
- Machine Learning often you often just supply features, and the *machine learns* complicated functions of the inputs

$$\hat{y} = f(x_1, x_2, x_3)$$

  - No need to specify nonlinear functions and interactions, a good machine learning model will figure those out with enough data

- Still need to feed the machine particular inputs (features, x's) to predict the outcome
- Still need to specify what is a good model (loss, ultimate objective)
- Difficult to interpret the final model (unlike linear regression, lots of features and much more complex)

Popular ML Algorithms (Supervised Learning)



Popular ML Algorithms (Supervised Learning)

- Tree Based Algorithms
  - Decision Trees
  - Random Forests & Boosting (Ensemble)
- Support Vector Machines (n < p)
- K-Nearest Neighbors
- Neural Networks/Deep Learning

- What one to choose? Depends on situation:
  - Should always have a baseline simple model in which to judge the benefit of more complicated models
  - Augmented Machine Learning (e.g. Data Robot) does standardized comparisons among many models given the same input data.

Additional ML Algorithms

- Unsupervised Learning (not predicting a known outcome, find similarity, compress data, feature engineering)
  - Clustering of similar data (k-means, hierarchical clustering)
    - Useful for outlier detection (not in a cluster means it is an outlier)
  - Mixture modelling/identifying latent traits
    - Natural Language Processing (NLP) of documents

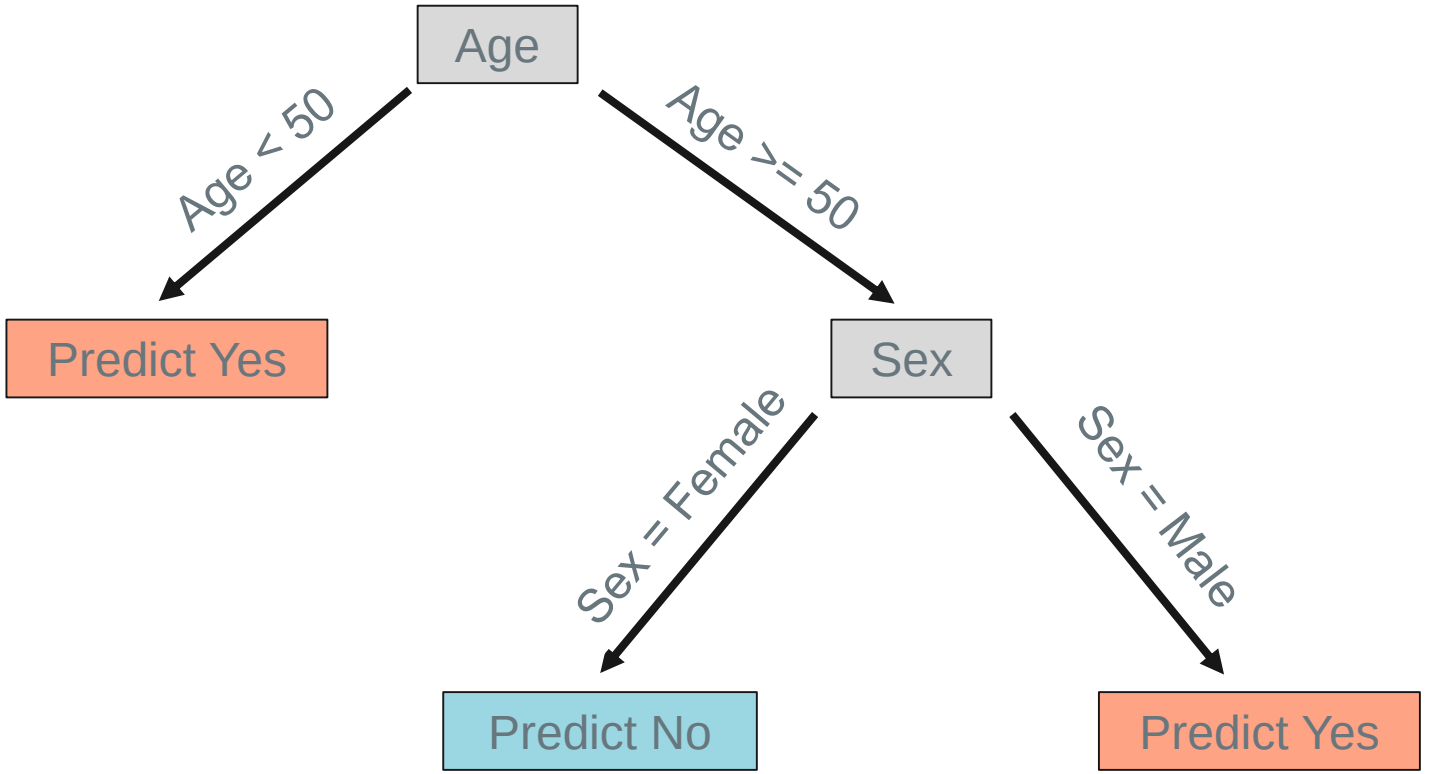
- Other Types of ML Algorithms
  - Recommender Systems (e.g. Netflix, Amazon Ads)
  - Reinforcement Learning (e.g. Multi-armed bandits, Self-Driving Cars)

Random Forests and Ensembling Methods

- Random Forests
  - Generate a bootstrapped sample of original data
  - Create a decision tree to make a prediction
  - Repeat many times
  - End prediction is the combination of the many individual trees

- Generalized Boosted Models
  - Estimate a base model
  - Calculate residuals
  - Train new model on residuals
  - Combine original model and new model
  - Repeat many times

Decision Trees



Example Using Python

- What we will be doing today

1) Load in data
2) Estimate a logistic & random forest model
3) Evaluate accuracy of those two models
4) Visualize differences in predictions

- Example predicting **obesity** using data from the Behavioral Risk Factor Survey
- Original data can be downloaded from https://health.data.ny.gov/Health/Behavioral-Risk-Factor-Surveillance-Survey-2015/rcr8-b3jj (I've only chosen a subset of variables.)
- Notebook available at https://github.com/hmsholdings/data-science-utils/tree/master/education/Intro_DataScience/MachineLearning_101

Loading in Data

In [1]:
```
#Loading in the libraries we will be using
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
import os

#Setting the working directory to where our data is stored
os.chdir(r'C:\Users\e009156\Documents\DataScience_Notes\Learning_Materials\DataSci_101\MachineLearning_101\Analysis')

#Reading in the CSV data
brfss_dat = pd.read_csv('Prepped_BRFSS2015.csv')

#A quick view of the first few rows of data
brfss_dat.head()
```

Out[1]:

	Obese_BMI	CurrentSmoker	SEX	MinActWeek	AgeMid
0	1	0	Male	120.0	70
1	0	0	Female	0.0	60
2	0	0	Male	336.0	70
3	0	0	Female	420.0	30
4	0	0	Female	300.0	60

This is the same data we used for Data Science 101 lesson

Estimating Two Models (Logistic & Random Forest)

In [2]:
```
#Estimating a logistic regression equation
#and a random forest model

#Changing sex to dummy variable, regression does not understand text
brfss_dat['Male'] = 1*(brfss_dat['SEX'] == 'Male')
ind_vars = ['Male','MinActWeek','AgeMid','CurrentSmoker']

logit_model = LogisticRegression(penalty='none', solver='newton-cg')
logit_model.fit(X = brfss_dat[ind_vars], y = brfss_dat['Obese_BMI'])

#setting the tree depth so it only have at max 5 leaves
#n_estimators is the number of trees
rf_model = RandomForestClassifier(n_estimators=100, max_depth=10, min_samples_leaf=50)
rf_model.fit(X = brfss_dat[ind_vars], y = brfss_dat['Obese_BMI'])
```

We specify the random forest has 100 trees, the maximum splits a tree can have (depth) is 10, and make it so we don't split anymore if a leaf has fewer than 50 observations.

Out[2]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=10, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=50, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Testing the Accuracy of Each Model

In [3]:
```
#Getting the predicted probability of obesity per our model
pred_probL = logit_model.predict_proba(X = brfss_dat[ind_vars])[::,1]
pred_probR = rf_model.predict_proba(X = brfss_dat[ind_vars])[::,1]

#Generating a confusion matrix, setting threshold to predict obese at 27%
th = 0.27
con_matL = pd.DataFrame(confusion_matrix(brfss_dat['Obese_BMI'], pred_probL > th),
                        columns=['Predict No','Predict Yes'], index=['Not Obese', 'Obese'])
con_matR = pd.DataFrame(confusion_matrix(brfss_dat['Obese_BMI'], pred_probR > th),
                        columns=['Predict No','Predict Yes'], index=['Not Obese', 'Obese'])

#The correct guesses are on the diagonal of the confusion matrix
accuracyL = (con_matL.iloc[0,0] + con_matL.iloc[1,1] ) / len(brfss_dat)
print("Accuracy Logit Model")
print("%.2f" % accuracyL)
print( con_matL )

accuracyR = (con_matR.iloc[0,0] + con_matR.iloc[1,1] ) / len(brfss_dat)
print("\nAccuracy Random Forest Model")
print("%.2f" % accuracyR)
print( con_matR )
```

Results on next slide!

Testing the Accuracy of Each Model

Accuracy Logit Model
0.56

	Predict No	Predict Yes
Not Obese	4661	3572
Obese	1366	1557

Accuracy Random Forest Model
0.58

	Predict No	Predict Yes
Not Obese	4704	3529
Obese	1133	1790

Based on a threshold of predicted 27%, the RF model is more accurate, and captures more obese individuals.

Applying Predictions to New Data

In [4]:
```
#Apply predictions to newdata

act = range(0,480)

new_dat = pd.DataFrame({'Male': 1, 'MinActWeek': act, 'AgeMid': 40, 'CurrentSmoker': 0})
new_dat['PredProbLogitMale'] = logit_model.predict_proba(new_dat[ind_vars])[::,1]
new_dat['PredProbRFMale'] = rf_model.predict_proba(new_dat[ind_vars])[::,1]
new_dat.head(10)
```

Out[4]:

	Male	MinActWeek	AgeMid	CurrentSmoker	PredProbLogitMale	PredProbRFMale
0	1	0	40	0	0.254304	0.285835
1	1	1	40	0	0.254141	0.285835
2	1	2	40	0	0.253979	0.285835
3	1	3	40	0	0.253816	0.278612
4	1	4	40	0	0.253653	0.278371
5	1	5	40	0	0.253491	0.278371
6	1	6	40	0	0.253328	0.278371
7	1	7	40	0	0.253166	0.276966
8	1	8	40	0	0.253004	0.276966
9	1	9	40	0	0.252841	0.276966

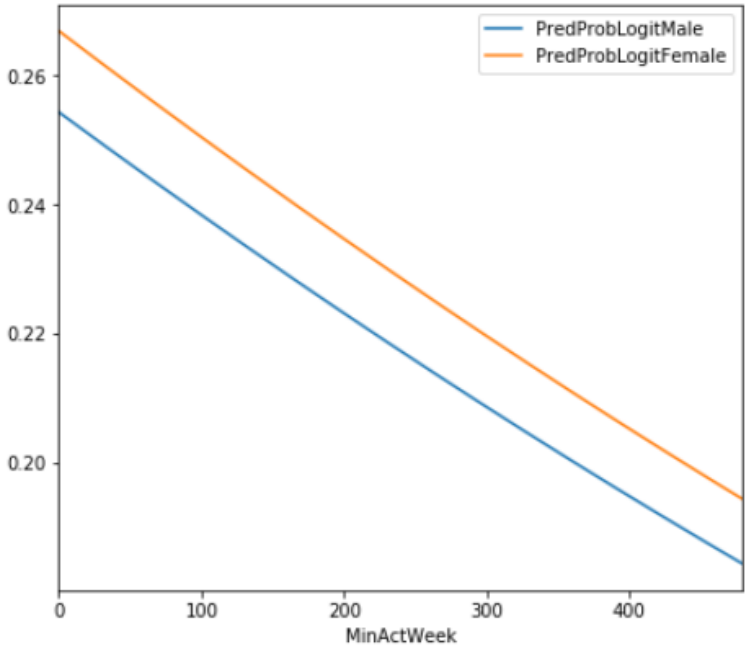
Random Forests predict males have a higher probability of being obese than logistic.

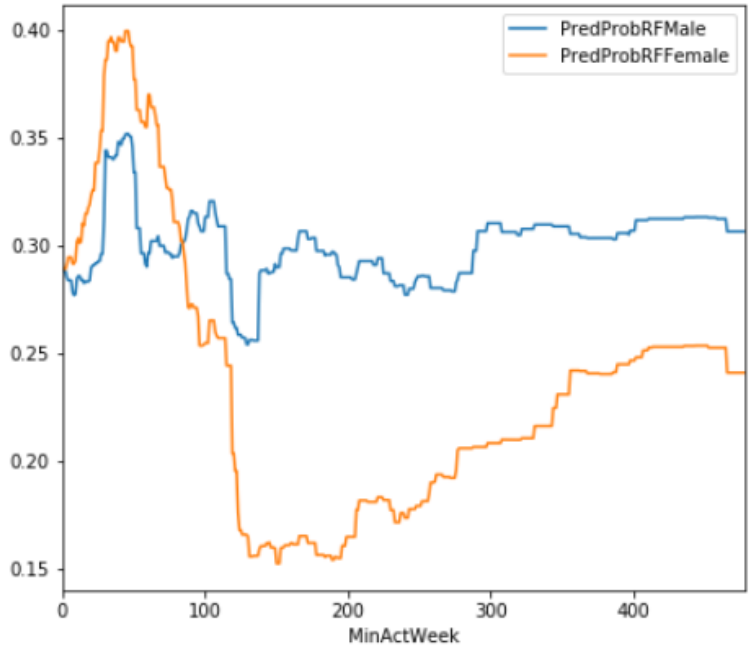
Logistic vs Random Forest Predictions

In [6]:
```
#Line graph comparing males to females for rf vs logit
new_dat['Male'] = 0
new_dat['PredProbLogitFemale'] = logit_model.predict_proba(new_dat[ind_vars])[::,1]
new_dat['PredProbRFFemale'] = rf_model.predict_proba(new_dat[ind_vars])[::,1]

import matplotlib.pyplot as plt
fig, axes = plt.subplots(ncols=2, figsize=[15,6]) #sharey=True to get them to have the same y axis
#Logit predictions, RF predictions
new_dat[['MinActWeek','PredProbLogitMale','PredProbLogitFemale']].plot(x='MinActWeek', ax = axes[0])
new_dat[['MinActWeek','PredProbRFMale','PredProbRFFemale']].plot(x='MinActWeek', ax= axes[1])
plt.show()
```

Random Forests are non-linear, and can find interactions between variables.





X-axis is activity per week, and y axis is the probability of being obese.

Future Topics

- How to validate predictions and benchmark different models on new samples.
- Incorporate different costs of false positives and false negatives.
- Show how to choose the best hyperparameters for Random Forest.

Questions?

Future Topics

Have requests?

Let me know!

Introduction to Data Science Course Outline

Andrew Wheeler, PhD, andrew.wheeler@hms.com

- Lesson 01: Data Science 101
- Lesson 02: Machine Learning 101
- Lesson 03: Evaluating Predictions
- Lesson 04: Intro Data Transformation in Python
- Lesson 05: Data Visualization 101
- Lesson 06: Feature Engineering
- Lesson 07: Missing Data
- Lesson 08: Big Data and Parallel Computing Intro
- Lesson 09: Dimension Reduction and Unsupervised Learning
- Lesson 10: High Cardinality (Many Categories)
- Lesson 11: Intro to Forecasting
- Lesson 12: Conducting Experiments

Machine Learning 101

Data Science Team

08/26/2020

Andrew Wheeler, PhD

andrew.wheeler@hms.com