imlplots

Julia Fried, Tobias Riebe, Christian Scholbeck

2018-04-11

imlplots: interpretable machine learning plots

`imlplots` is an R package that provides an interactive Shiny dashboard for three kinds of Interpretable Machine Learning (IML) plots

- Partial Dependence Plots (PDP)
- Individual Conditional Expectation (ICE) plots
- Accumulated Local Effect (ALE) plots

Installation

The package can be installed directly from github with devtools

```
# install.packages("devtools")
devtools::install_github('juliafried/imlplots')
library(imlplots)
```

Quickstart

You can fit classification and regression problems from the `mlr` package and analyse possible interaction effects in a Shiny dasbhoard.

For quickstart we take the popular Boston Housing data, where we want to predict the median housing price in Boston.

```
print(summarizeColumns(boston)[, -c(5, 6, 7)], digits = 4)
```

```
##       name    type na     mean       min     max nlevs
## 1     crim numeric  0   3.6135   0.00632  88.976     0
## 2       zn numeric  0  11.3636   0.00000 100.000     0
## 3    indus numeric  0  11.1368   0.46000  27.740     0
## 4     chas  factor  0       NA  35.00000 471.000     2
## 5      nox numeric  0   0.5547   0.38500   0.871     0
## 6       rm numeric  0   6.2846   3.56100   8.780     0
## 7      age numeric  0  68.5749   2.90000 100.000     0
## 8      dis numeric  0   3.7950   1.12960  12.127     0
## 9      rad  factor  0       NA  17.00000 132.000     9
## 10     tax numeric  0 408.2372 187.00000 711.000     0
## 11 ptratio numeric  0  18.4555  12.60000  22.000     0
## 12   black numeric  0 356.6740   0.32000 396.900     0
## 13   lstat numeric  0  12.6531   1.73000  37.970     0
## 14    medv numeric  0  22.5328   5.00000  50.000     0
```

For using `imlplots` Shiny dashboard, three input arguments need to be specified

- `data` - the input data

- `task`- the learning task
- `models` - one or several trained models

We create a regression task with `medv` as target variable. The task structure is determined by `mlr` package.

```
boston.task = makeRegrTask(data = boston, target = "medv")
```

The `imlplots` dashboard allows the comparison of multiple learning algorithms, therefore we fit two different models - first a random forest and second a GLM.

```
rf.mod = train("regr.randomForest", boston.task)
glm.mod = train("regr.glm", boston.task)
```

The input for the Shiny app is a list of learners.

```
mod.list = list(rf.mod, glm.mod)
```

Now the Shiny app can be used.

```
imlplots(data = boston, task = boston.task, models = mod.list)
```

Code for Copy & Paste

```
boston.task = makeRegrTask(data = boston, target = "medv")

rf.mod = train("regr.randomForest", boston.task)
glm.mod = train("regr.glm", boston.task)
mod.list = list(rf.mod, glm.mod)

imlplots(data = boston, task = boston.task, models = mod.list)
```

Further Examples

IML Plots for Regression Tasks

To show how you can use the `imlplots` Shiny app for regression tasks we use fire data, where the burned area of forests due to fires should be analyzed.

```
print(summarizeColumns(fire)[, -c(5, 6, 7)], digits = 4)
```

```
##     name    type na      mean  min    max nlevs
## 1  month  factor  0        NA  1.0  184.0    12
## 2    day  factor  0        NA 54.0   95.0     7
## 3   FFMC numeric  0  90.64468 18.7   96.2     0
## 4    DMC numeric  0 110.87234  1.1  291.3     0
## 5     DC numeric  0 547.94004  7.9  860.6     0
## 6    ISI numeric  0   9.02166  0.0   56.1     0
## 7   temp numeric  0  18.88917  2.2   33.3     0
## 8     RH integer  0  44.28820 15.0  100.0     0
## 9   wind numeric  0   4.01760  0.4    9.4     0
## 10  rain numeric  0   0.02166  0.0    6.4     0
## 11  area numeric  0  12.84729  0.0 1090.8     0
```

The target variable is `area`, which is between 0.00 and 1090.84 ha.

```
summary(fire$area)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.52   12.85    6.57 1090.84
```

We create a regression task with target variable `area`.

```
fire.task = makeRegrTask(data = fire, target = "area")
```

Next we train several `mlr` models and save them in a list of models. Note: The order in your model list will determine the model order in the Shiny dashboard.

```
fire.rf = train("regr.randomForest", fire.task)
fire.glm = train("regr.glm", fire.task)
fire.lm = train("regr.lm", fire.task)

mod.list = list(fire.rf, fire.glm, fire.lm)
```

No we can open the `imlplots` Shiny app.

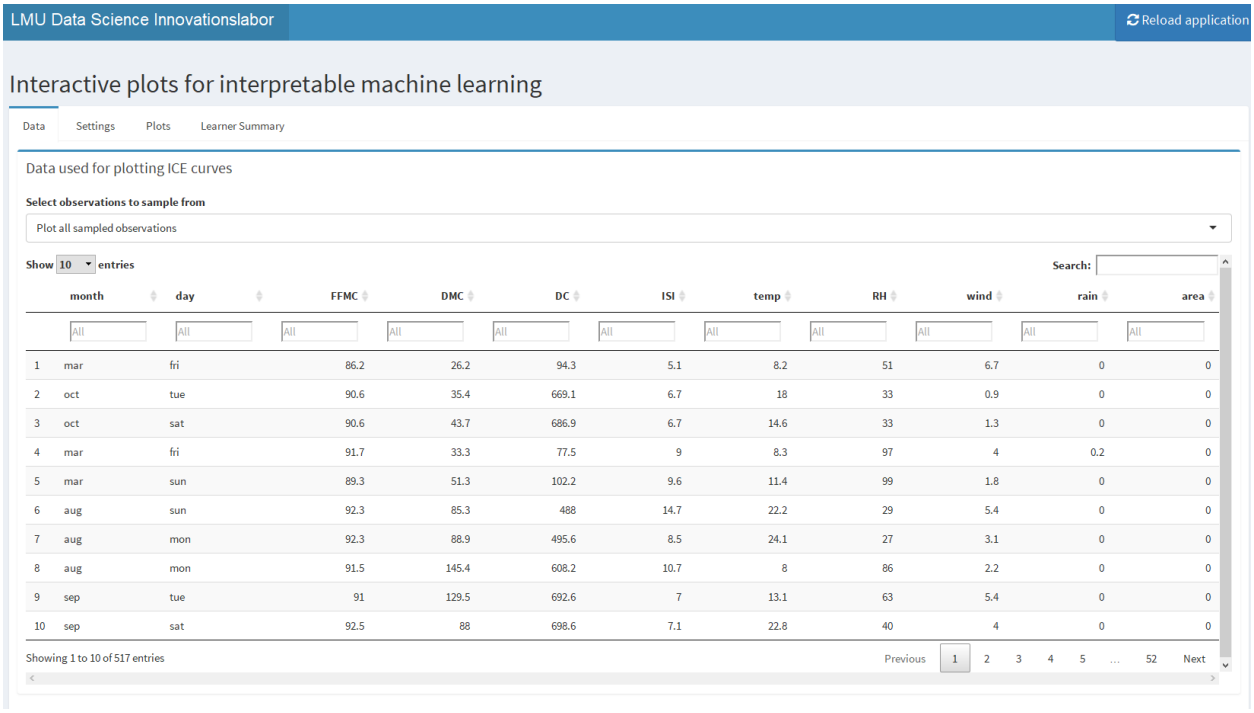
```
imlplots(data = fire, task = fire.task, models = mod.list)
```

The Shiny dashboard contains four tabs

- Data
- Settings
- Plots
- Learner Summary

Data

The `Data` tab shows your input data. This data is taken to generate IML plots. If you want to check how changes in the data effect your plot, you can simply filter in the `Data` tab.

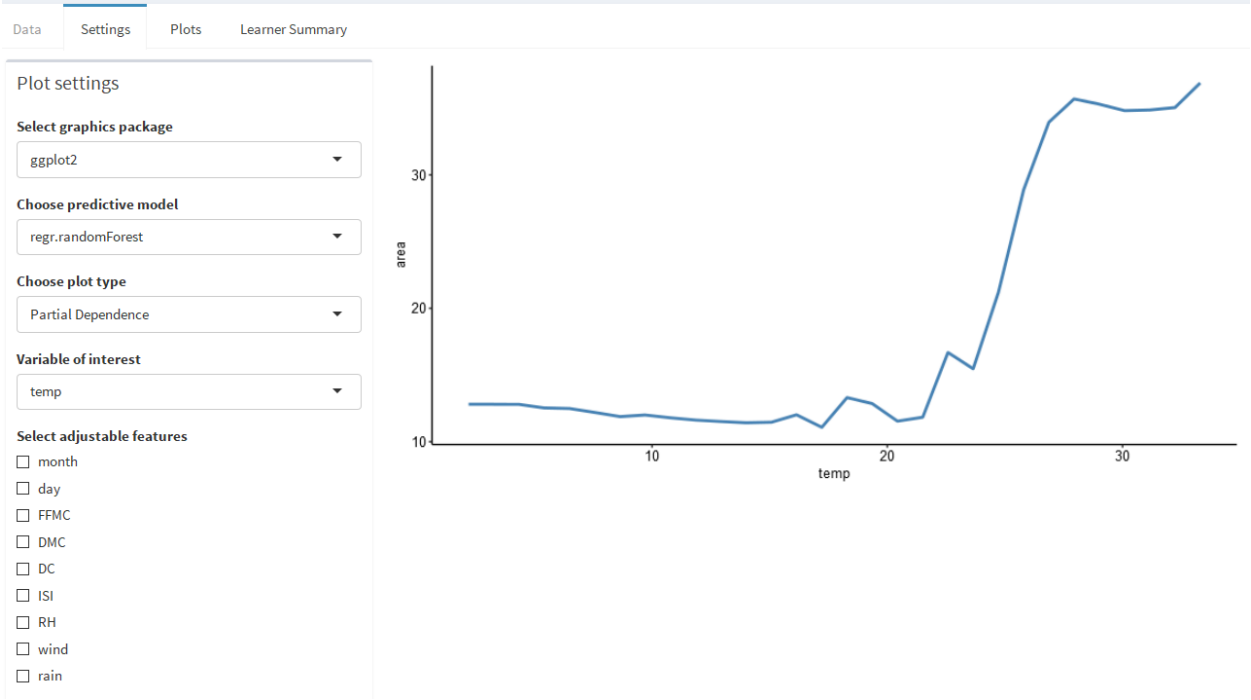


For filtering two options are given

1. `Plot all sampled observations`: In this setting you can filter via the filters beneath the column titles and all rows will be used for plotting.
2. `Plot indiviudal observations`: In this setting after using the filters, you have to manually select specific rows.

Settings

The next tab `Settings` contains all possible plot settings and the selected IML plot.



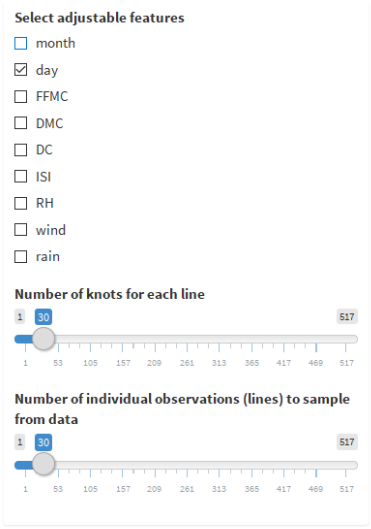
There are various settings

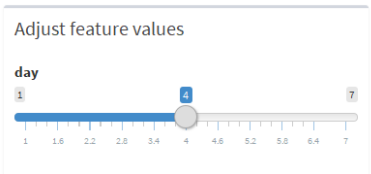
1. `Select graphics package`: You can select the graphics package - we offer `ggplot2` and `plotly`. Use `ggplot2` if your computer is not the fastest one.
2. `Choose predictive model`: Choose one of your fitted models. The order in the dropdown is the order of your list.
3. `Choose plot type`: We offer PDP, ICE and ALE plots If you select ICE plot, you will get a new selection field. Possible are `centered` and `regular` ICE plots
4. `Variable of interest`: This dropdown will determine the x-axis of your plot and will determine the effect that is plotted

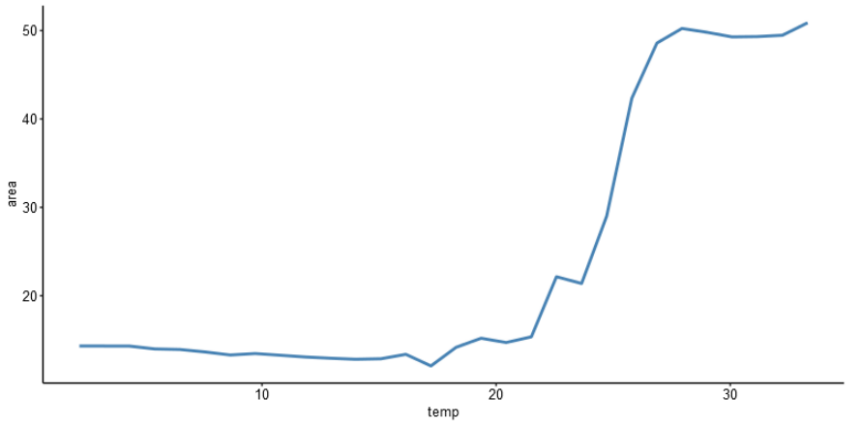
PDP

On the right side of the dashboard page, the selected plot is shown.

To check out effects, you can turn on `Select adjustable features`. This option allows you to set one of the variables to a specifc value.



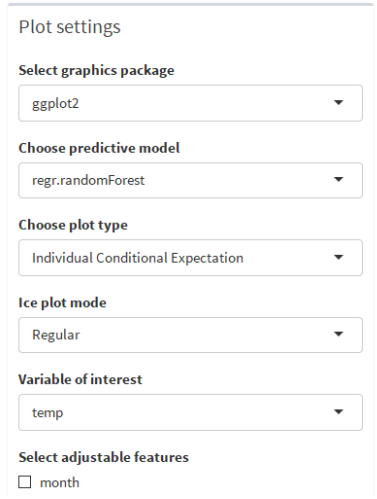


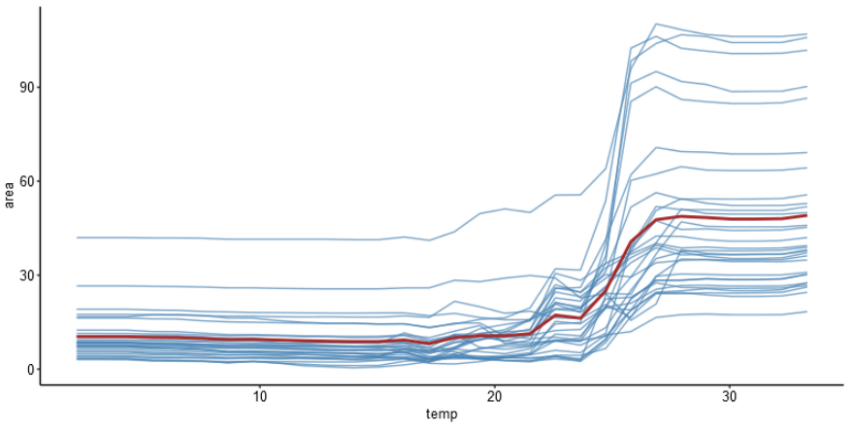


It is also possible to change the number of knots and lines (individual observations) with the shown sliders.

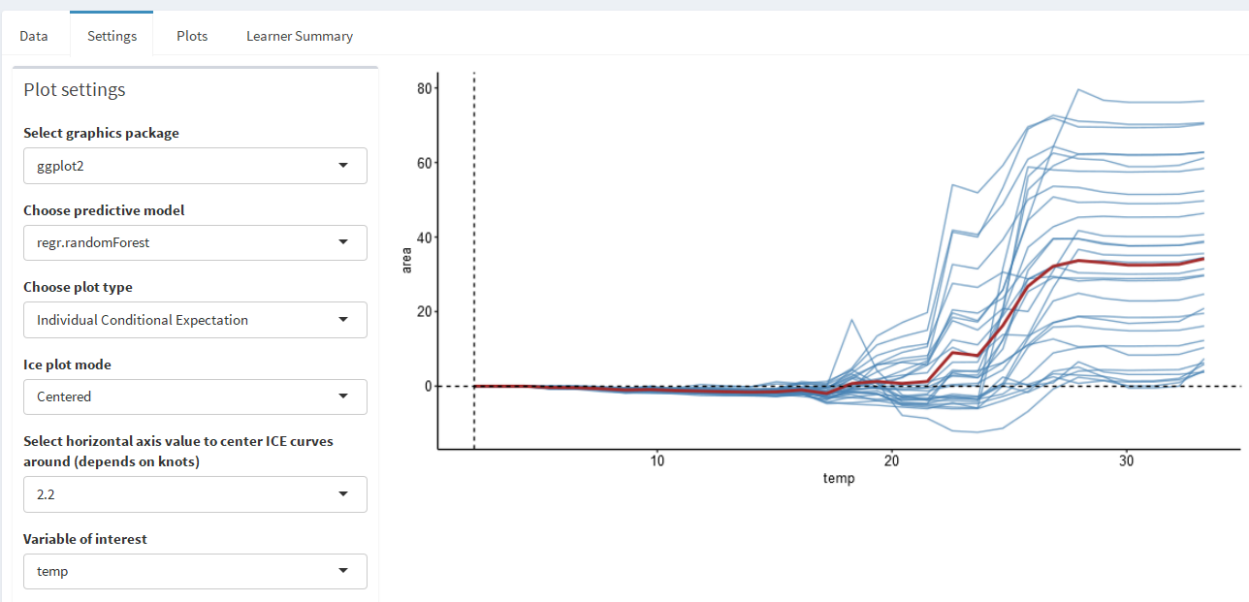
ICE Plot

The ICE plot contains all sampled, individual observations in blue. The red line is from PDP.



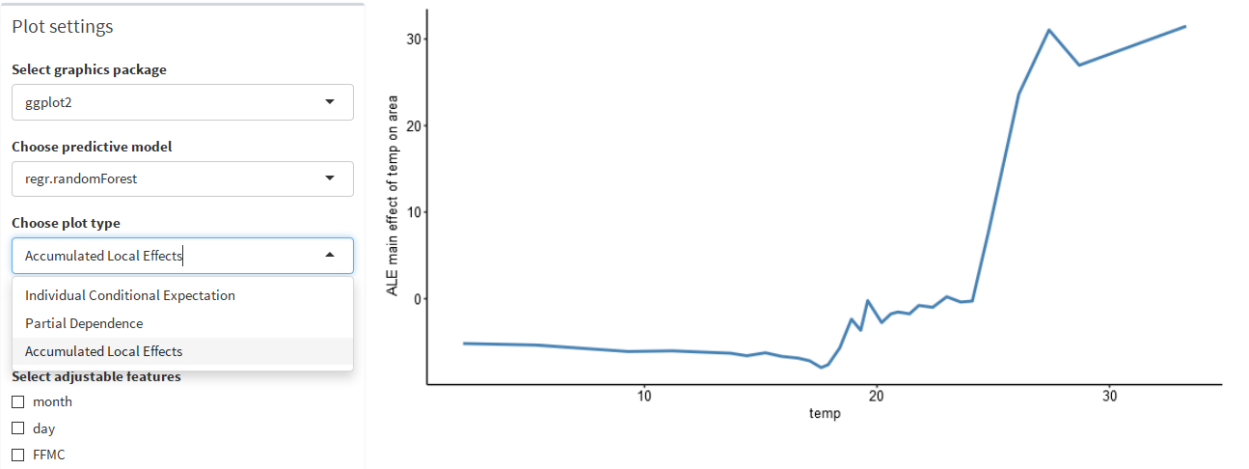


As described above, you can select between `Regular` and `Centered` ICE plots.

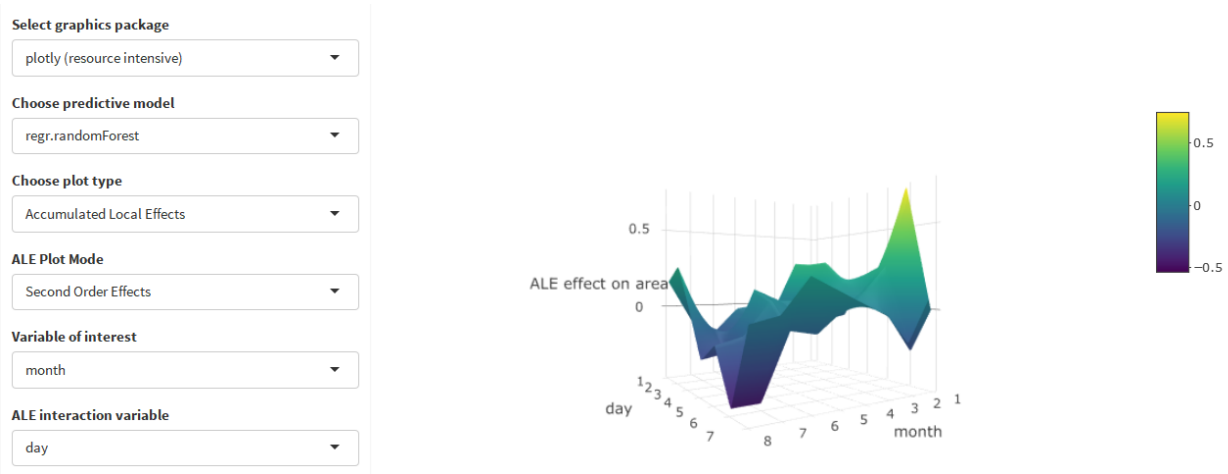


ALE Plot

The ALE plot can be selected, too. Please keep in mind, that the ALE plot has a different y-axis than the PDP and ICE plot.



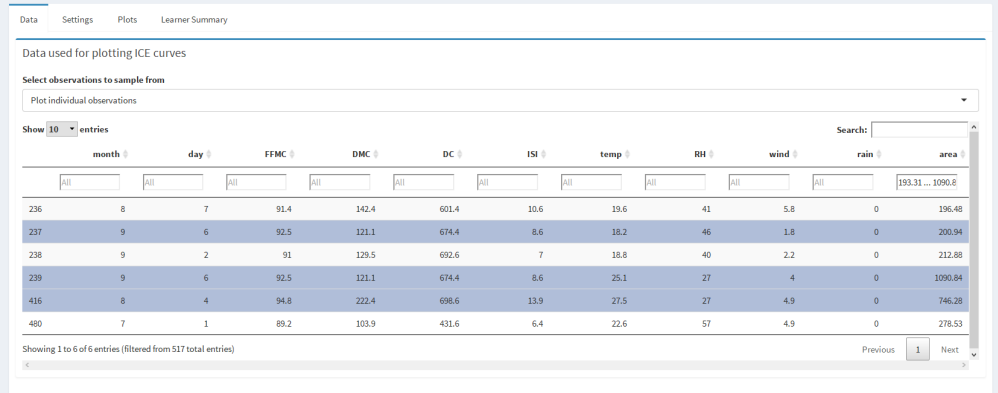
For ALE plots you can swith between two `ALE Plot Modes`. The `Main Effets` mode allows you to select one variable of interest and shows its interaction effect. The `Second Order Effects` setting allows to select another `ALE interaction variable` and therefore shows the effect for this extra variable too. If you select `plotly` as graphics package, the second order effects ALE plot will be a 3D plot.



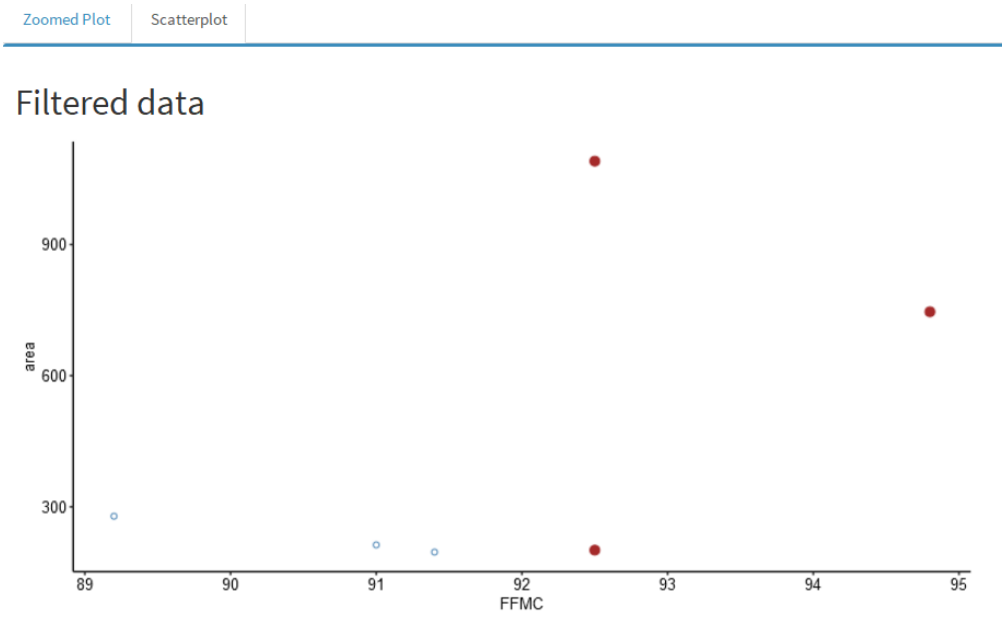
Plots

The third tab `Plots` shows the selected IML plot in full screen via the sub-tab `Zoomed plot`. The sub-tab `Scatterplot` shows the filtered and unfiltered scatterplot between the `variable of interest` and the `target` variable of the model.

In the `Data` tab we filtered for a high value of burned area and selected three individual observations.



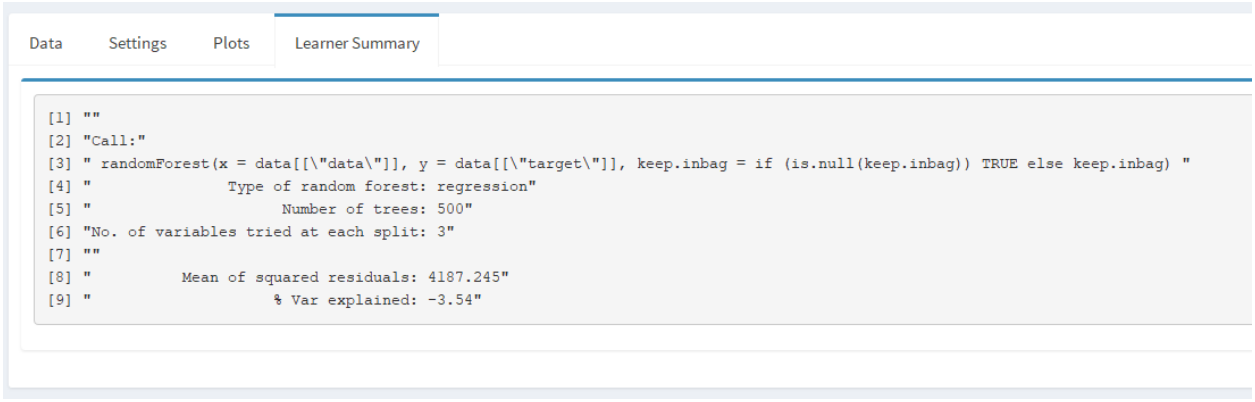
The filtered data scatterplot shows the selected high area values and also the three individual observations (in red).



The unfiltered data scatterplot shows all data points and also the three individual observations (in red).

Learner Summary

The fourth tab `Learner Summary` shows the currently selected learner summary. If you want to see another summary, you have to select another model in the `Settings` tab.



Code for Copy & Paste

```
library(imlplots)

fire.rf = train("regr.randomForest", fire.task)
fire.glm = train("regr.glm", fire.task)
fire.lm = train("regr.lm", fire.task)

mod.list = list(fire.rf, fire.glm, fire.lm)

imlplots(data = fire, task = fire.task, models = mod.list)
```

IML Plots for Classification Tasks

For the classification example only the differences to the regression example will be explained. We use the titanic data set, where the aim is to predict the survival chance.

```
print(summarizeColumns(titanic)[, -c(5, 6, 7)], digits = 4)
```

```
##        name    type na    mean      min    max nlevs
## 1    Pclass integer  0  2.2949   1.0000    3.0     0
## 2  Survived integer  0  0.3820   0.0000    1.0     0
## 3       Sex  factor  0      NA 466.0000  843.0     2
## 4       Age numeric  0 29.5032   0.1667   80.0     0
## 5     Sibsp integer  0  0.4989   0.0000    8.0     0
## 6     Parch integer  0  0.3850   0.0000    9.0     0
## 7      Fare numeric  0 33.2811   0.0000  512.3     0
## 8  Embarked  factor  0      NA   2.0000  914.0     4
## 9    farePp numeric  0 20.5090   0.0000  512.3     0
## 10     deck  factor  0      NA   1.0000 1014.0     9
## 11 portside  factor  0      NA 110.0000 1059.0     3
```

Again we create a task and fit a model.

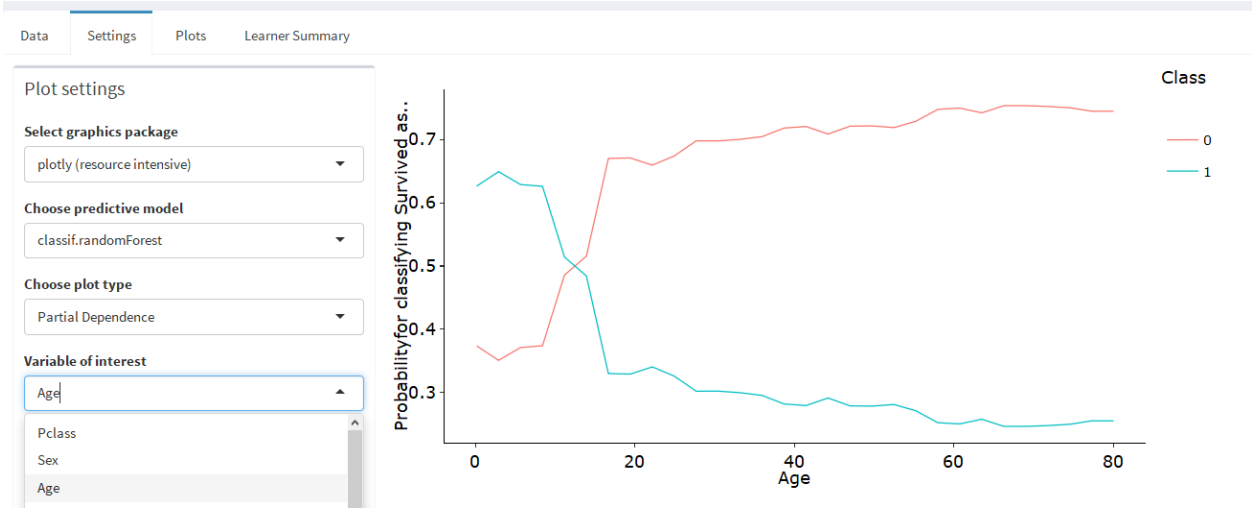
```
library(imlplots)

titanic.task = makeClassifTask(data = titanic, target = "Survived")
titanic.rf = train("classif.randomForest", titanic.task)
```

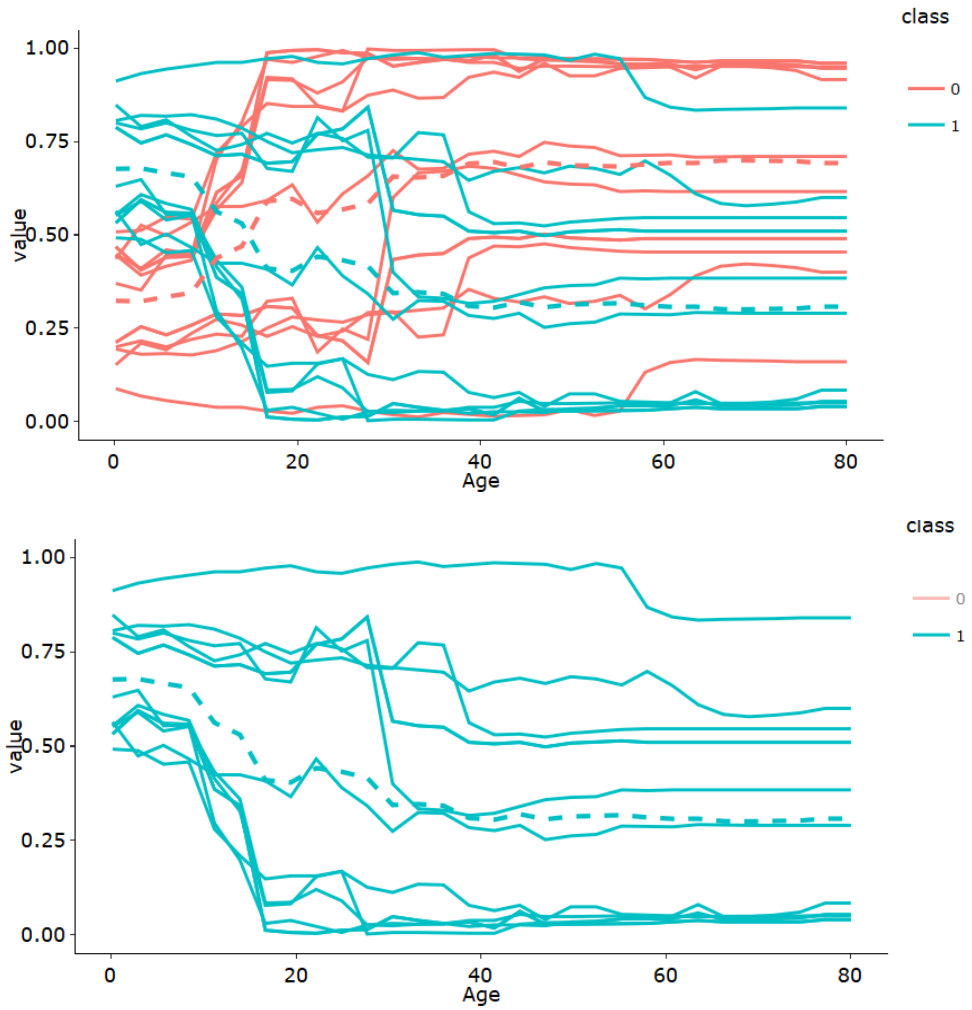
Next we open the Shiny dashboard.

```
imlplots(data = titanic, task = titanic.task, titanic.rf)
```

This time it is useful to select `plotly` in the `Select graphics package` dropdown.



This allows you to deselect single classes to increase the visability of individual lines, which is very useful for ICE plot.



Code for Copy & Paste

```
library(imlplots)

titanic.task = makeClassifTask(data = titanic, target = "Survived")
titanic.rf = train("classif.randomForest", titanic.task)

imlplots(data = titanic, task = titanic.task, titanic.rf)
```

References

- References