

Logistic Regression

Data Set

```
attrition <- attrition %>% mutate_if(is.ordered, factor, order = F)
attrition.h2o <- as.h2o(attrition)
```

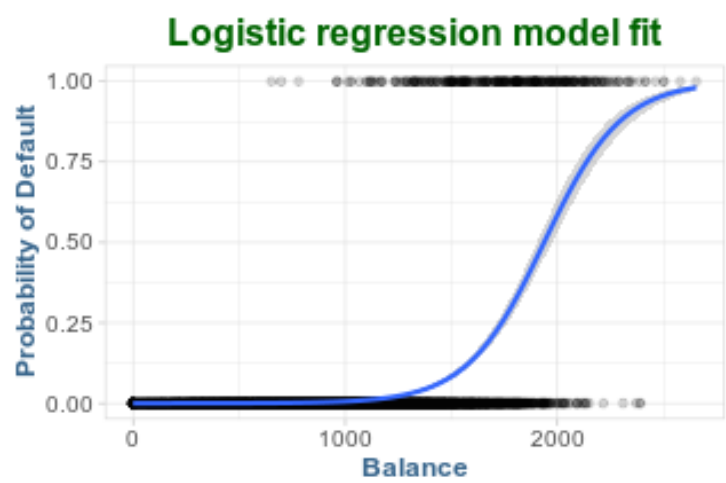
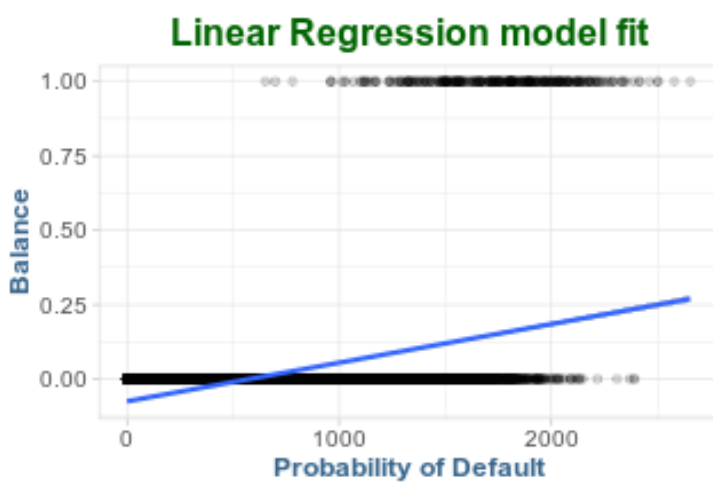
Overview

Linear Regression vs Logistic Regression

```
p1 <- ISLR::Default %>%
  mutate(prob = ifelse(default == "Yes", 1, 0)) %>%
  ggplot(aes(balance, prob)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "lm") +
  ggtitle("Linear Regression model fit") +
  ylab("Balance") + xlab("Probability of Default")

p2 <- ISLR::Default %>%
  mutate(prob = ifelse(default == "Yes", 1, 0)) %>%
  ggplot(aes(balance, prob)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model fit") +
  xlab("Balance") + ylab("Probability of Default")

gridExtra::grid.arrange(p1, p2, nrow = 1)
```



Create training (70%) and test (30%) tests.

```
set.seed(123)

churn_split <- initial_split(attrition,
                             prop = 0.7, strata = "Attrition")

churn_train <- training(churn_split)
churn_test  <- testing(churn_split)
```

Simple Logistic Regression

Fit two generalized linear models to predict attrition.

```
model1 <- glm(Attrition ~ MonthlyIncome, family = "binomial",
              data = churn_train)

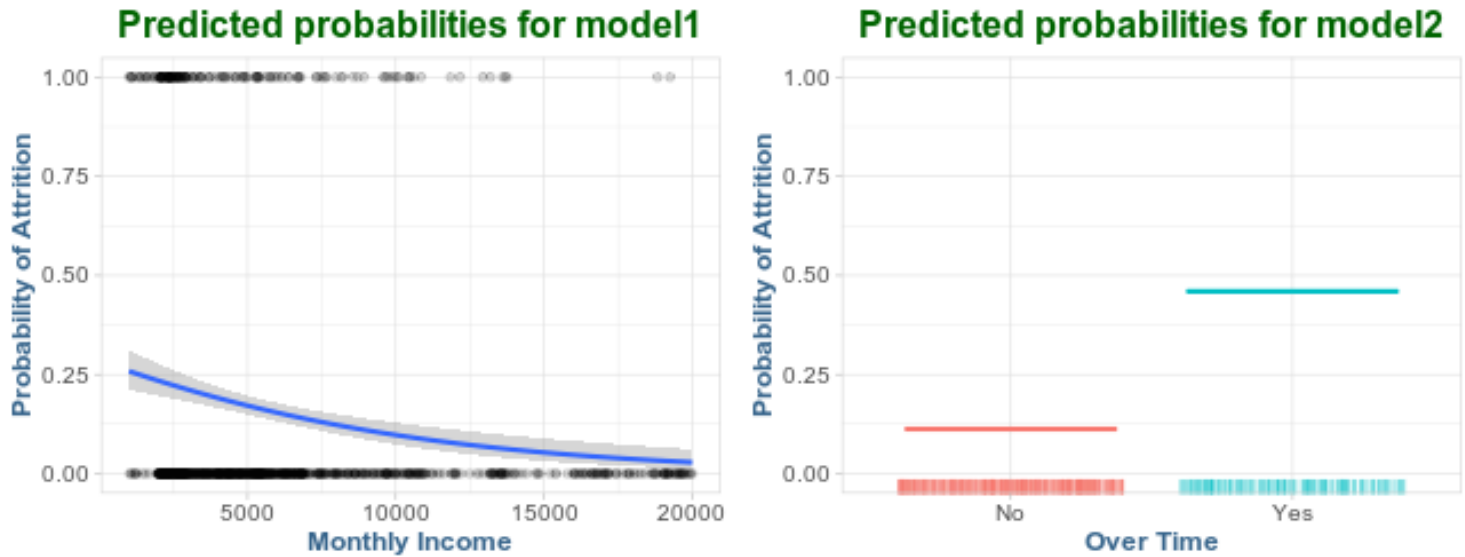
model2 <- glm(Attrition ~ OverTime, family = "binomial",
              data = churn_train)

churn_train2 <- churn_train %>% mutate(prob = ifelse(Attrition == "Yes", 1, 0))
churn_train2 <- broom::augment(model2, churn_train2) %>% mutate(.fitted = exp(.fitted))

p1 <- ggplot(churn_train2, aes(MonthlyIncome, prob)) +
  geom_point(alpha = 0.15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Predicted probabilities for model1") +
  xlab("Monthly Income") +
  ylab("Probability of Attrition")

p2 <- ggplot(churn_train2, aes(OverTime, .fitted, color = OverTime)) +
  geom_boxplot(show.legend = F) +
  geom_rug(sides = "b", position = "jitter", alpha = 0.2, show.legend = F) +
  ggtitle("Predicted probabilities for model2") +
  xlab("Over Time") +
  scale_y_continuous("Probability of Attrition", limits = c(0, 1))

gridExtra::grid.arrange(p1, p2, nrow = 1)
```



Model Diagnostics

```
tidy(model1)
```

```
# A tibble: 2 x 5
  term      estimate std.error statistic    p.value
  <chr>      <dbl>    <dbl>    <dbl>    <dbl>
1 (Intercept) -0.924    0.155     -5.96 0.00000000259
2 MonthlyIncome -0.000130 0.0000264    -4.93 0.0000000836
```

```
tidy(model2)
```

```
# A tibble: 2 x 5
  term      estimate std.error statistic    p.value
  <chr>      <dbl>    <dbl>    <dbl>    <dbl>
1 (Intercept)  -2.18    0.122    -17.9 6.76e-72
2 OverTimeYes   1.41    0.176     8.00 1.20e-15
```

```
exp(coef(model1))
```

```
(Intercept) MonthlyIncome
0.3970771    0.9998697
```

```
exp(coef(model2))
```

```
(Intercept) OverTimeYes
0.1126126    4.0812121
```

```
confint(model1)
```

Waiting for profiling to be done...

2.5 % 97.5 %

```
(Intercept)    -1.2267754960 -0.61800619157
MonthlyIncome  -0.0001849796 -0.00008107634
```

```
confint(model2)
```

```
Waiting for profiling to be done...
```

```
              2.5 %      97.5 %
(Intercept) -2.430458 -1.952330
OverTimeYes  1.063246  1.752879
```

Multiple Logistic Regression

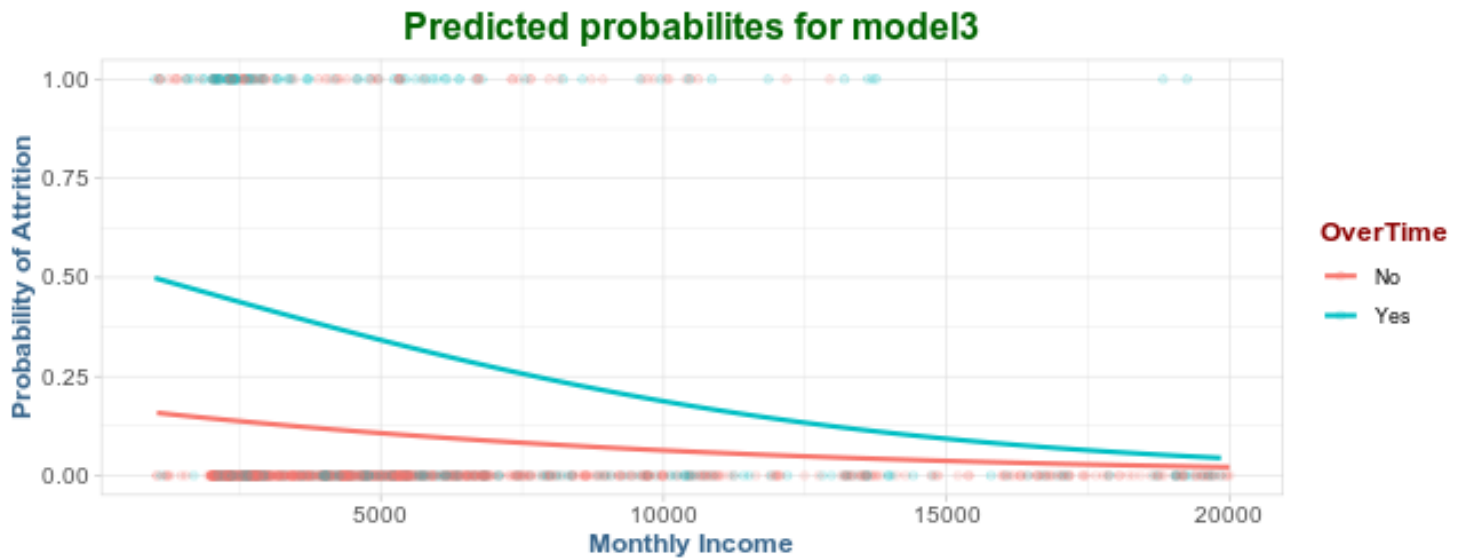
```
model3 <- glm(
  Attrition ~ MonthlyIncome + OverTime,
  family = "binomial",
  data = churn_train
)
```

```
tidy(model3)
```

```
# A tibble: 3 x 5
  term          estimate std.error statistic  p.value
<chr>         <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  -1.43        0.176      -8.11 5.25e-16
2 MonthlyIncome -0.000139 0.0000270    -5.15 2.62e- 7
3 OverTimeYes   1.47        0.180       8.16 3.43e-16
```

```
churn_train3 <- churn_train %>% mutate(prob = ifelse(Attrition == "Yes", 1, 0))
churn_train3 <- broom::augment(model3, churn_train3) %>% mutate(.fitted = exp(.fitted))
```

```
ggplot(churn_train3, aes(MonthlyIncome, prob, color = OverTime)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial"), se = F) +
  ggtitle("Predicted probabilities for model3") +
  xlab("Monthly Income") +
  ylab("Probability of Attrition")
```



Assessing Model Accuracy

```
set.seed(123)

cv_model1 <- train(
  Attrition ~ MonthlyIncome,
  data = churn_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

set.seed(123)

cv_model2 <- train(
  Attrition ~ MonthlyIncome + OverTime,
  data = churn_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

set.seed(123)

cv_model3 <- train(
  Attrition ~ .,
  data = churn_train,
```

```

method = "glm",
family = "binomial",
trControl = trainControl(method = "cv", number = 10)
)

# extract out of sample performance measure

summary(
  resamples(
    list(
      model_1 = cv_model1,
      model_2 = cv_model2,
      model_3 = cv_model3
    )
  )
)$statistics$Accuracy

```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
model_1	0.8349515	0.8349515	0.8365385	0.8388478	0.8431373	0.8446602	0
model_2	0.8349515	0.8349515	0.8365385	0.8388478	0.8431373	0.8446602	0
model_3	0.8365385	0.8495146	0.8792476	0.8757893	0.8907767	0.9313725	0

```

# predicted class
pred_class <- predict(cv_model3, churn_train)

# create confusion matrix
confusionMatrix(
  data = relevel(pred_class, ref = "Yes"),
  reference = relevel(churn_train$Attrition, ref = "Yes")
)

```

Confusion Matrix and Statistics

	Reference	
Prediction	Yes	No
Yes	93	25
No	73	839

Accuracy : 0.9049
 95% CI : (0.8853, 0.9221)
 No Information Rate : 0.8388
 P-Value [Acc > NIR] : 0.000000000536

 Kappa : 0.6016

McNemar's Test P-Value : 0.000002057257

Sensitivity : 0.56024
 Specificity : 0.97106
 Pos Pred Value : 0.78814
 Neg Pred Value : 0.91996
 Prevalence : 0.16117
 Detection Rate : 0.09029
 Detection Prevalence : 0.11456
 Balanced Accuracy : 0.76565

'Positive' Class : Yes

No-information rate

```
table(churn_train$Attrition) %>% prop.table()
```

No	Yes
0.838835	0.161165

Basically, this is saying if we just predicted "No" for every instance we would have 83.8% accuracy.

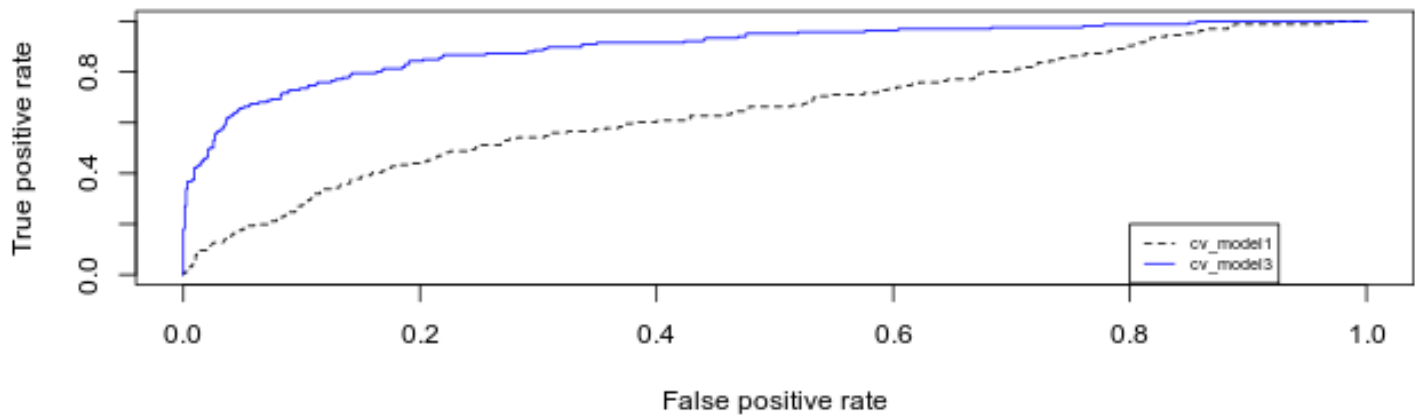
```
# Compute predicted probabilities
m1_prob <- predict(cv_model1, churn_train, type = "prob")$Yes
m3_prob <- predict(cv_model3, churn_train, type = "prob")$Yes

# Compute AUC metrics for cv_model1 and cv_model3
perf1 <- prediction(m1_prob, churn_train$Attrition) %>%
  performance(measure = "tpr", x.measure = "fpr")

perf2 <- prediction(m3_prob, churn_train$Attrition) %>%
  performance(measure = "tpr", x.measure = "fpr")

# Plot ROC curves for cv_model1 and cv_model3
plot(perf1, col = "black", lty = 2)
plot(perf2, col = "blue", add = T)

legend(0.8, 0.2, legend = c("cv_model1", "cv_model3"),
  col = c("black", "blue"), lty = 2:1, cex = 0.6)
```

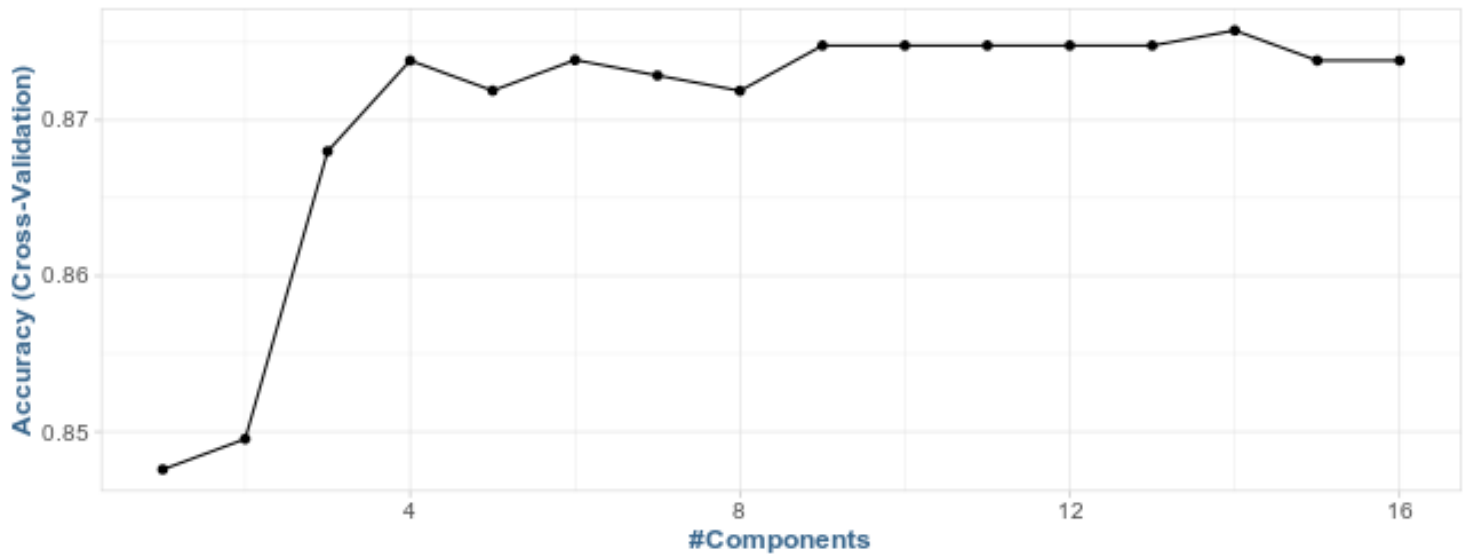


```
set.seed(123)
cv_model_pls <- train(
  Attrition ~ .,
  data = churn_train,
  method = "pls",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10),
  preProcess = c("zv", "center", "scale"),
  tuneLength = 16
)

cv_model_pls$bestTune

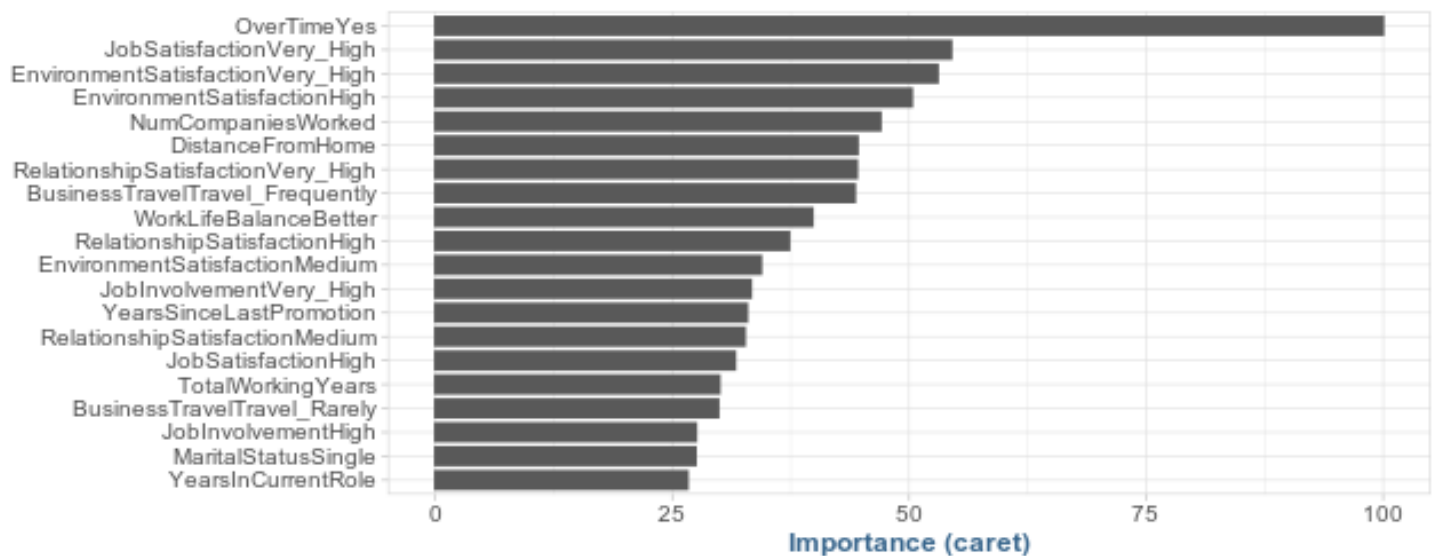
      ncomp
14      14

ggplot(cv_model_pls)
```

Feature Interpretation

```
vip::vip(cv_model3, num_features = 20)
```



```
pred.fun <- function(object, newdata) {
  Yes <- mean(predict(object, newdata, type = "prob")$Yes)
  as.data.frame(Yes)
}

p1 <- pdp::partial(cv_model3, pred.var = "OverTime", pred.fun = pred.fun ) %>%
  autoplot(rug = T) + ylim(c(0, 1))
```

```

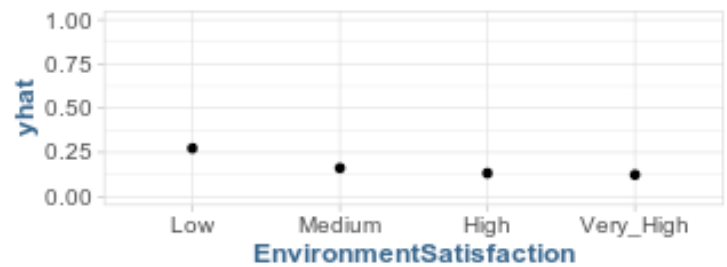
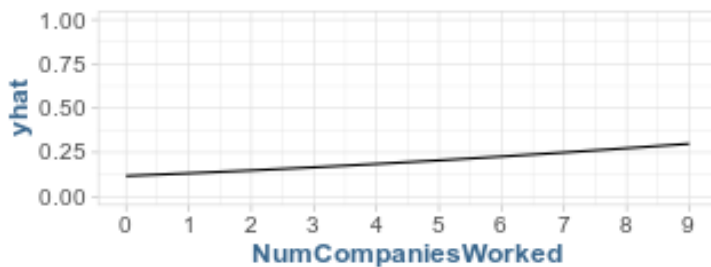
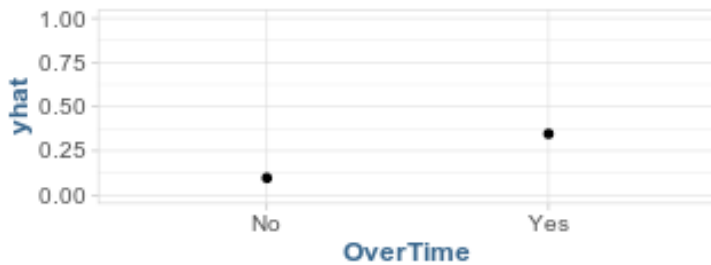
p2 <- pdp::partial(cv_model3, pred.var = "JobSatisfaction", pred.fun = pred.fun) %>%
  autoplot() + ylim(c(0, 1))

p3 <- pdp::partial(cv_model3, pred.var = "NumCompaniesWorked", pred.fun = pred.fun, gr = 10) %>%
  autoplot() + scale_x_continuous(breaks = 0:9) + ylim(c(0, 1))

p4 <- pdp::partial(cv_model3, pred.var = "EnvironmentSatisfaction", pred.fun = pred.fun) %>%
  autoplot() + ylim(c(0, 1))

grid.arrange(p1, p2, p3, p4, nrow = 2)

```



```

# clean up
rm(list = ls())

```