MONASH University

ETX2250/ETF5922: Data Visualization and Analytics

k Nearest Neighbours

Lecturer: *Lauren Kennedy*

Department of Econometrics and Business Statistics

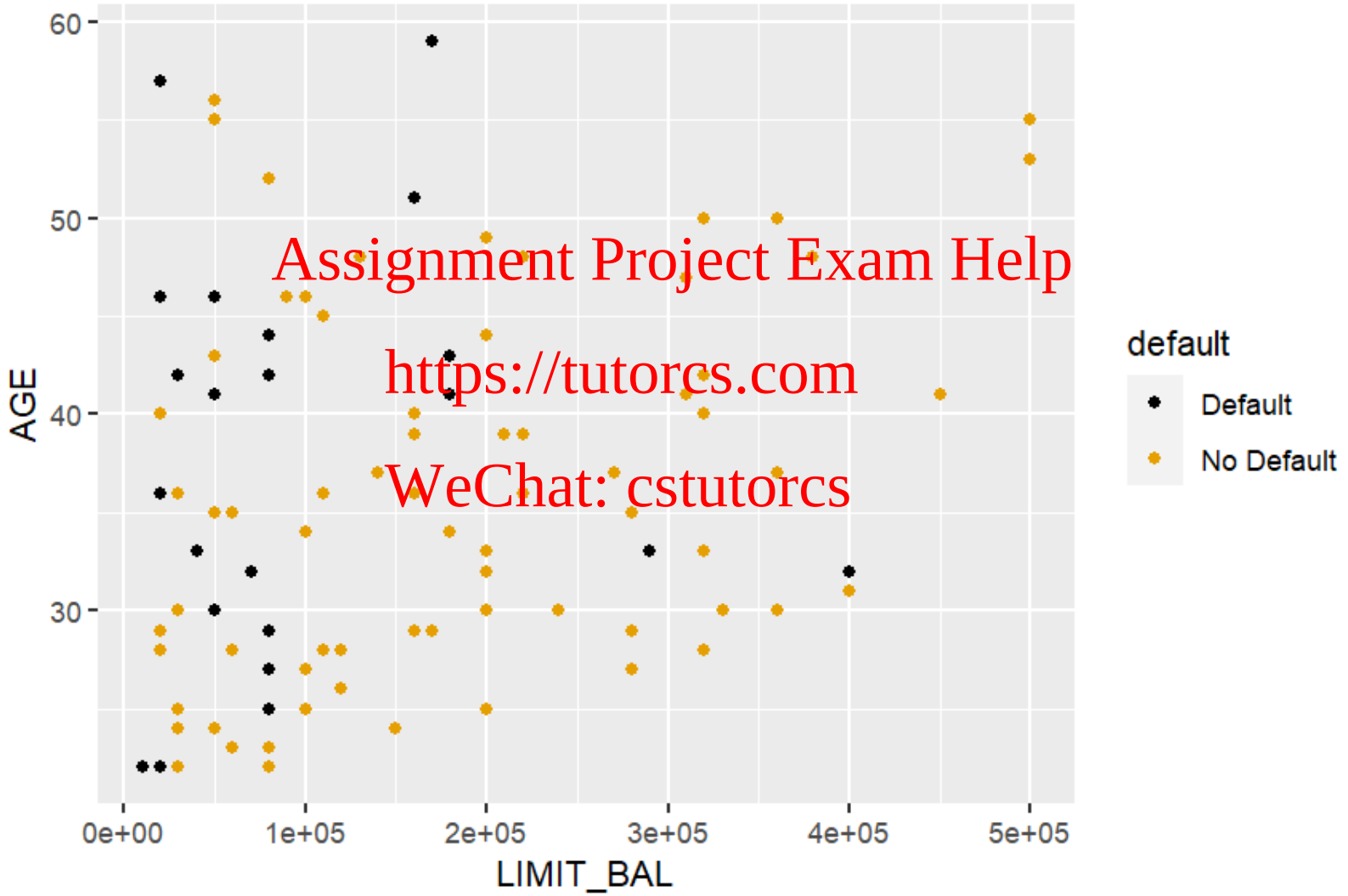
lauren.kennedy1@monash.edu

Week 11

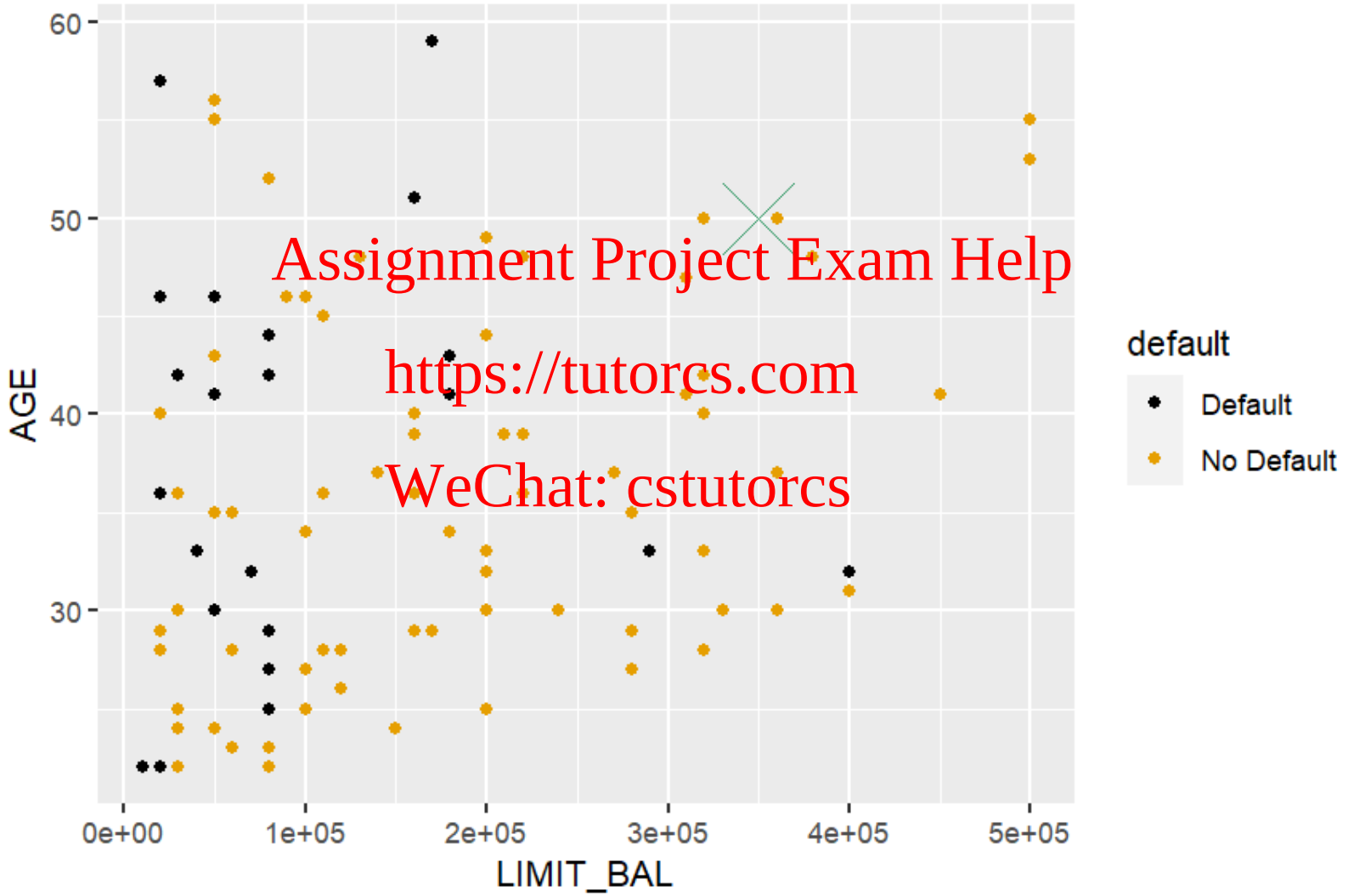
Nearest Neighbours

- Recall that we discussed smoothing by nearby points when we discussed density plots
- Now we will use nearest neighbours in the context of classification.
- Illustration with 2 predictors since these are easy to understand.
- Data is a subsample of 100 observations from the Credit default data.

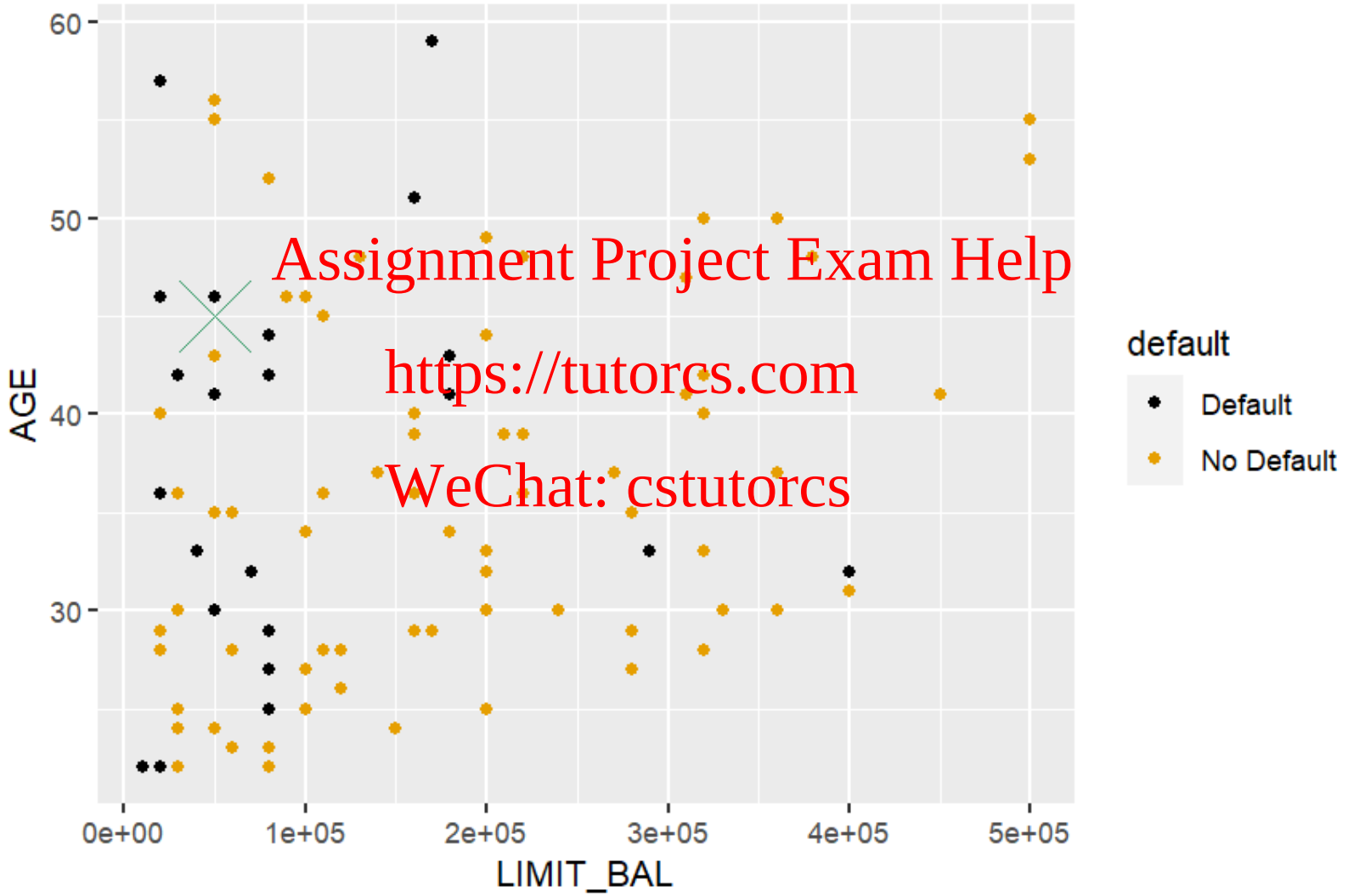
Credit Data



Default or not?



Default or not?



Basic Idea

- For the first point (limit 350000, age 50) most nearby points in the training data are not defaults (orange).
  - Predict no default
- For the second point (limit 50000, age 45) most nearby points in the training data are defaults (black).
  - Predict default

More formally

- Concentrate on the k nearest points, also known as the k Nearest Neighbours or k-NN.
- Let $\mathcal{N}_{\mathbf{x},k}$ be the set of k training points closest to $\mathbf{x}$.
- The predicted probability of Default is

$$\frac{1}{k}\sum_{i\in\mathcal{N}_{\mathbf{x},k}} I(y_i = \text{Default})$$

What do we mean by near?

- Recall the distance metrics from week 7, which would be appropriate here?
- kNN requires many distance estimates - we need to calculate the distance between each point that we need to classify and all of the points that we have observed.
- Euclidean distances is computationally cheap to calculate so this is most common (and what we use!)
- Remember the Euclidean distance between $u = \{u_1, u_2, \ldots, u_m\}$ and $v = \{v_1, v_2, \ldots, v_m\}$ is

$$\sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \ldots (u_m - v_m)^2}$$

Non-parametric

- kNN is a non-parametric method (linear regression is a parametric method)
- This means that unlike linear regression, we do not assume a particular form of relationship between the predictors

$$x_1, x_2, \ldots, x_m$$

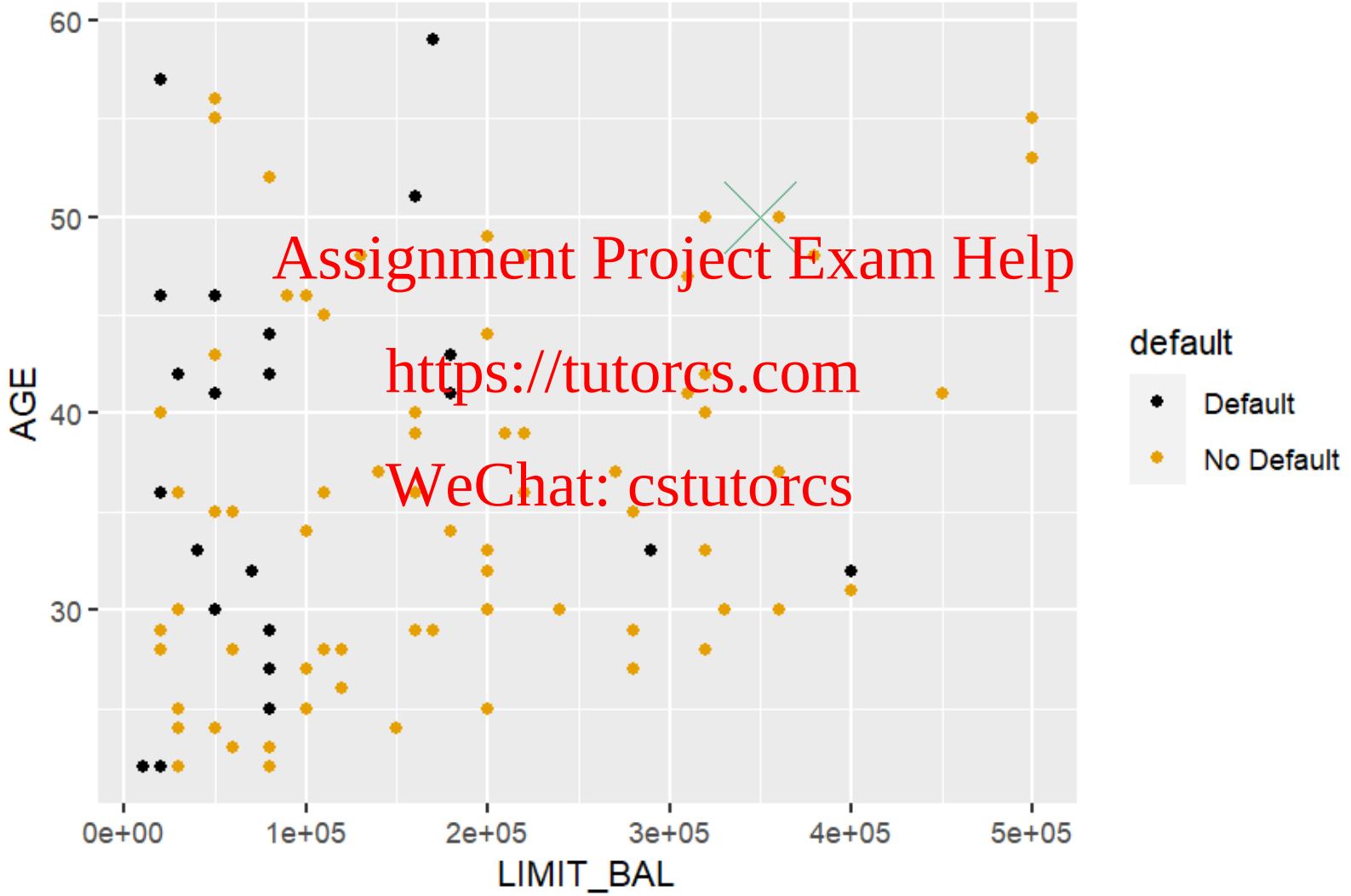
and the outcome.

- Does NOT mean there are no assumptions at all. k nearest neighbours assumes that points that are close by in space are more similar.

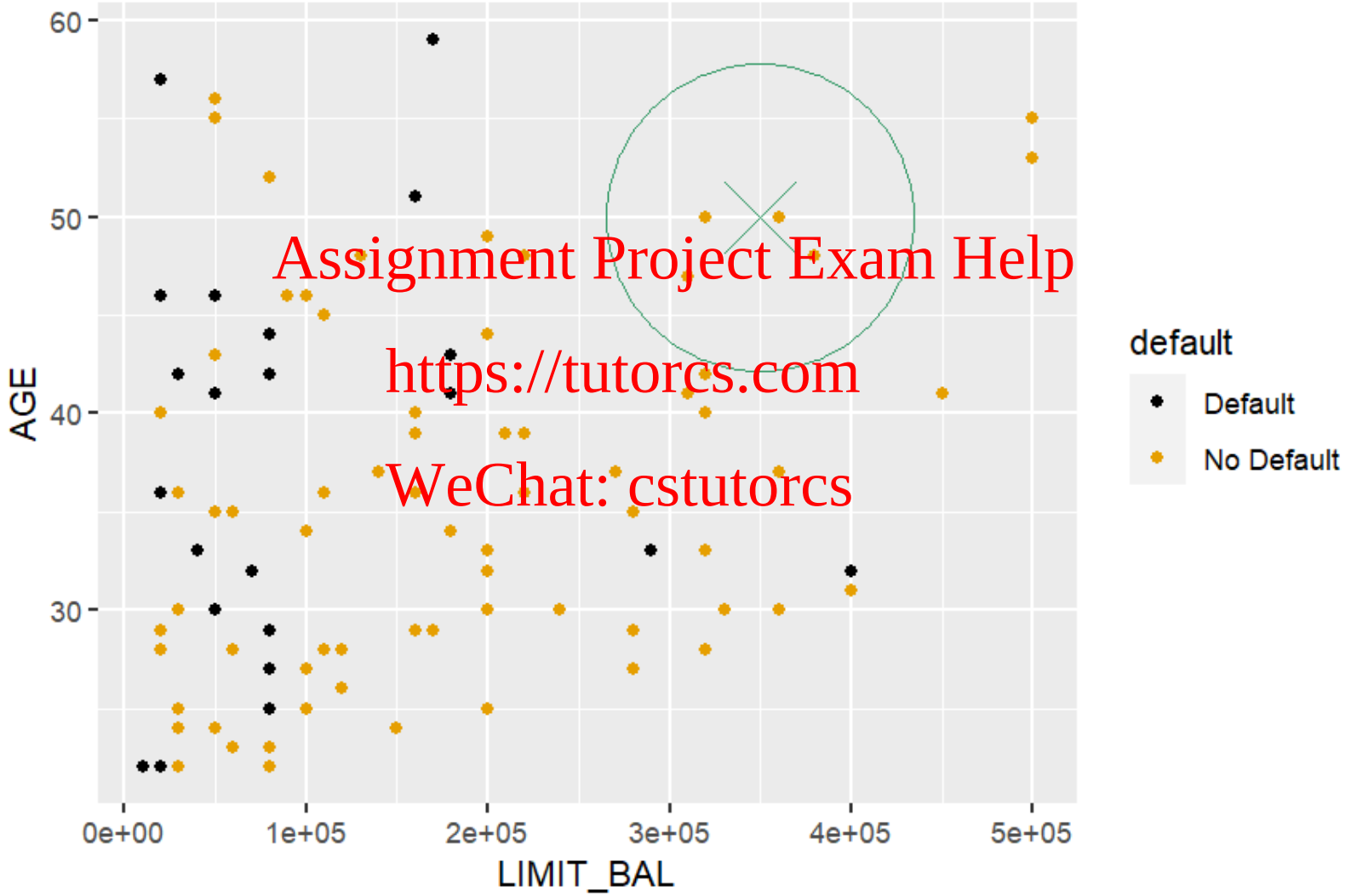
Your turn:

Thinking of the ethics of data based decision making, what is one benefit of a simple algorithm like kNN?

First example (k=4)



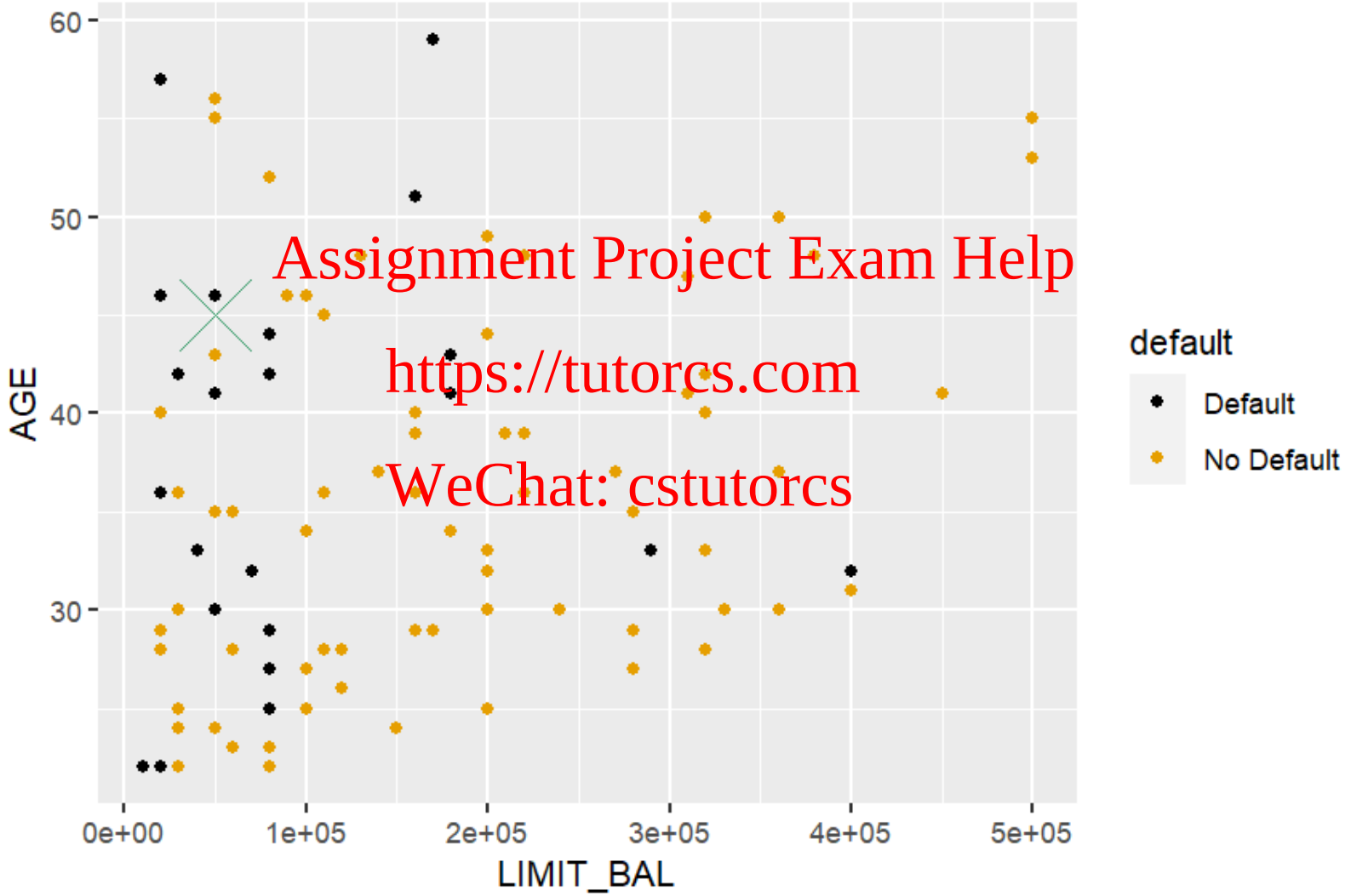
First example (k=4)



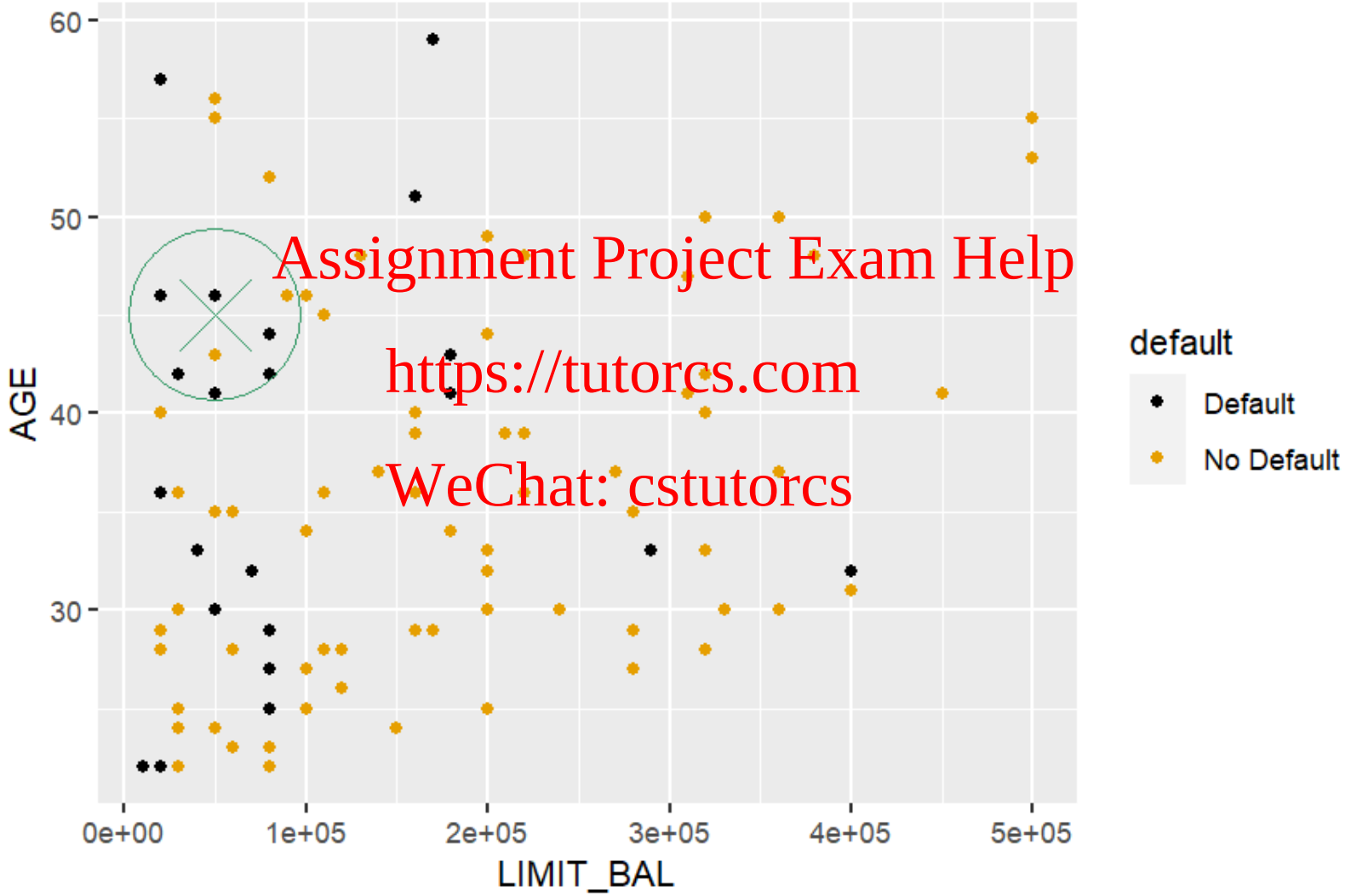
Prediction

- The four nearest neighbours are all *Non-defaults*
- Therefore the predicted default probability for an individual with limit of 350000 and age 50 is zero.
- We predict this individual does not default.

Second example (k=4)



Second example (k=4)



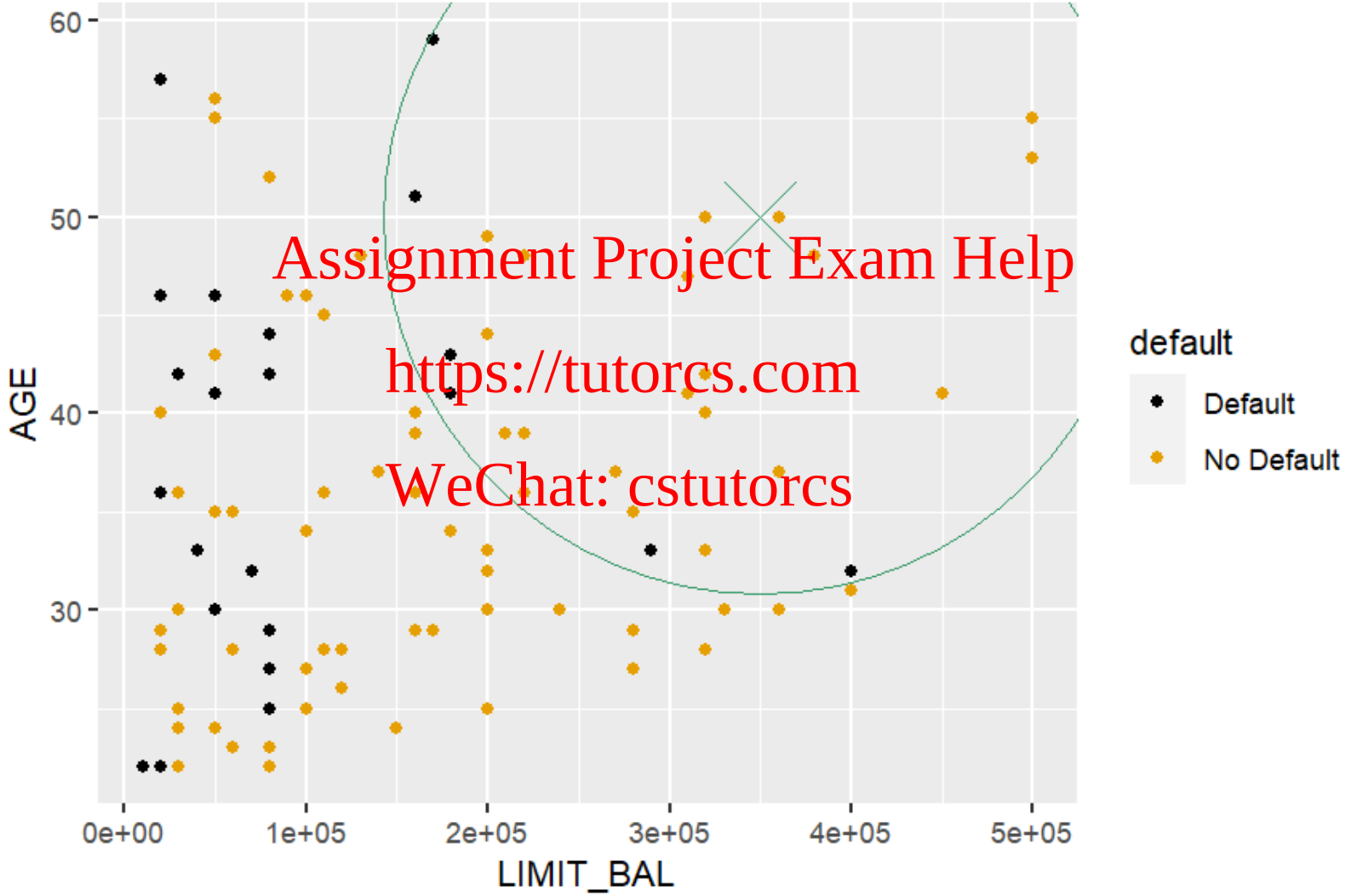
Prediction

- The four nearest neighbours include three *Defaults* and one *Non-default*
- Therefore the predicted default probability for an individual with limit of 50000 and age 45 is 0.75.
- Using a threshold of 0.5, we predict this individual does default.
- In practice a different threshold can be used.

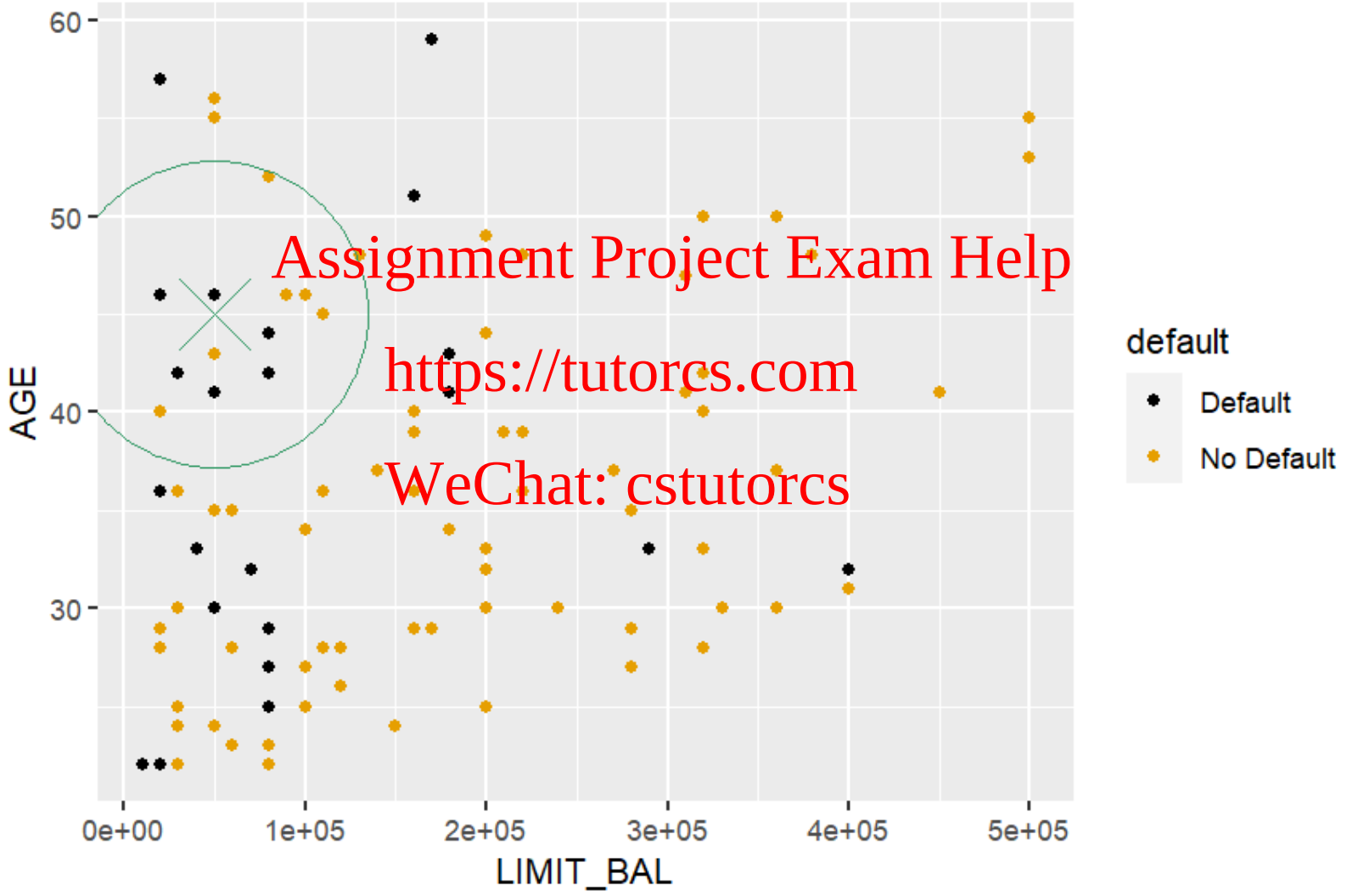
Your Turn

- Consider the case where $k = 10$
- What is the predicted probability of default for the first example (350000 limit 50 years age)?
- What is the predicted probability of default for the second example (50000 limit 45 years age)?

First example (k=10)



Second example (k=10)



Answers

- What is the predicted probability of default for the first example (350000 limit 50 years age)?
  - The predicted probability of default is 0.
- What is the predicted probability of default for the second example (50000 limit 45 years age)?
  - The predicted probability of default is 0.6.

Decision Boundary

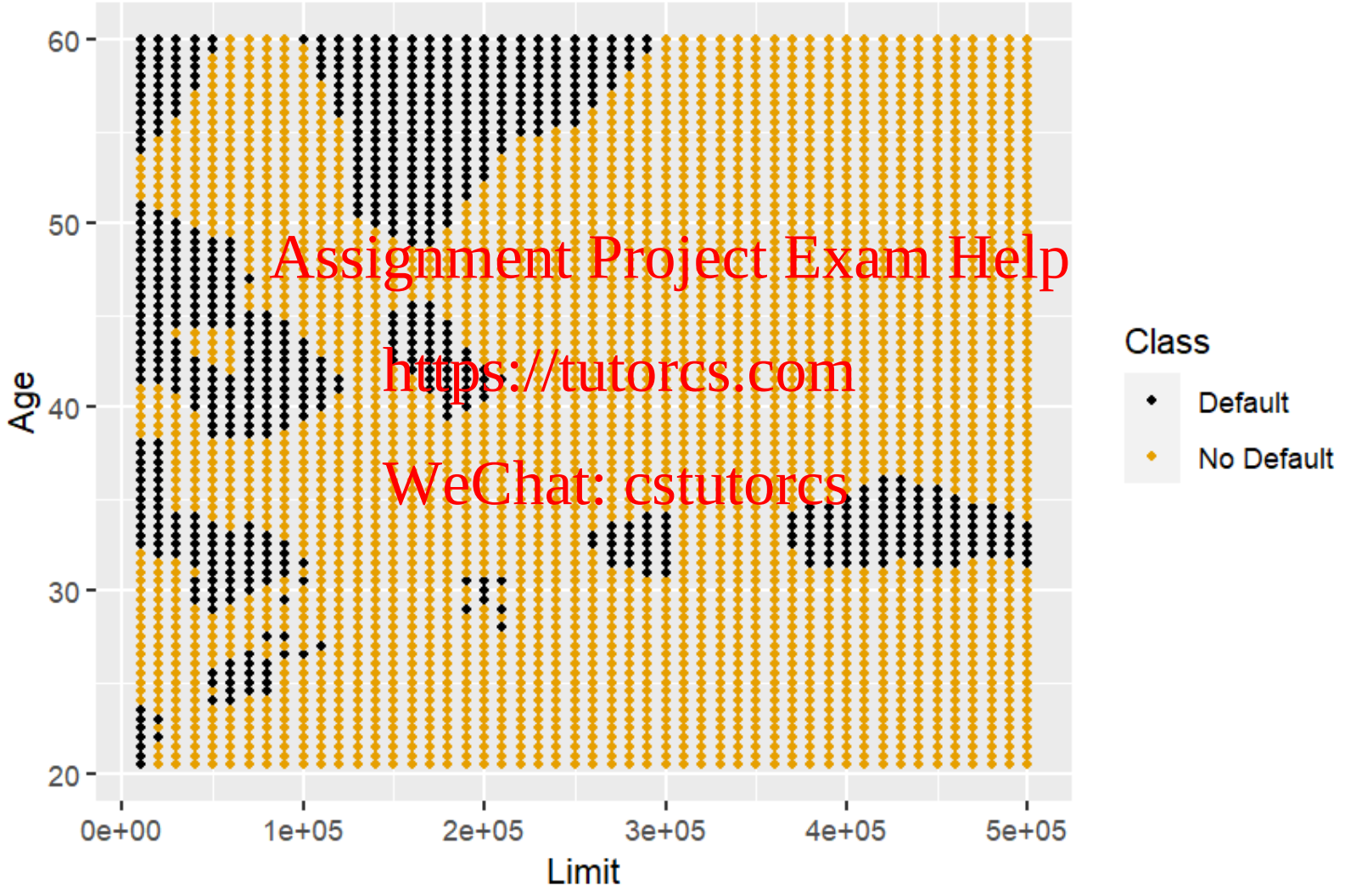
Data again

```
grid<-Credit[,c(2,6)]
colnames(grid)<-c('Limit','Age')
grid%>%
  as_tibble()%>%
  add_column(Class=as.factor(pull(Credit,25)))%>%
  ggplot(aes(x=Limit,y=Age,col=Class))+geom_point()+scale_color_colorblind()
```

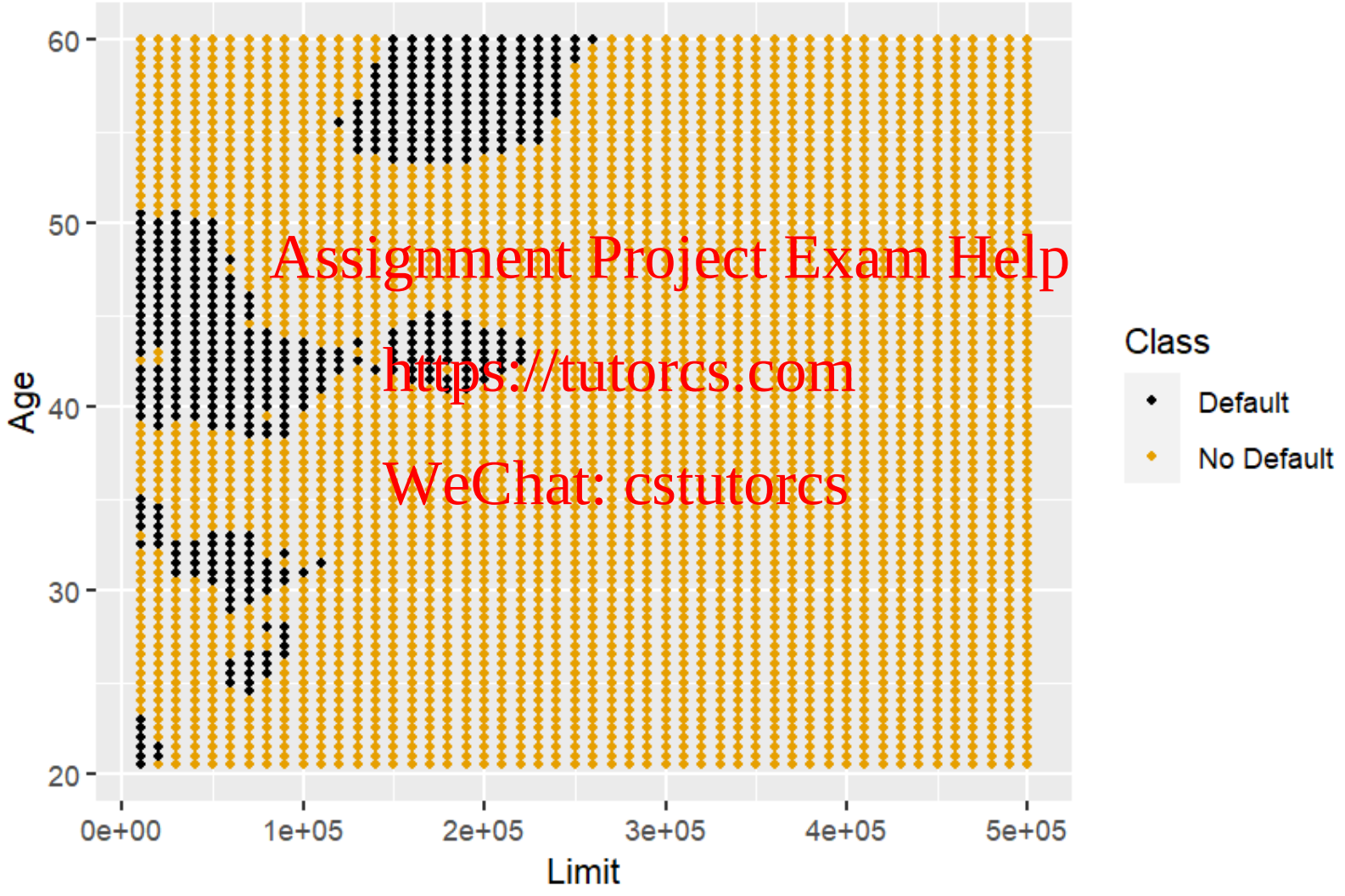
Decision Boundaries

- For the next few slides a grid of evaluation points has been created.
- If k-NN predicts "Default" the grid point is colored black.
- If k-NN predicts "No default" the grid point is colored yellow.
- Plotting these gives an idea of the *decision boundaries*.

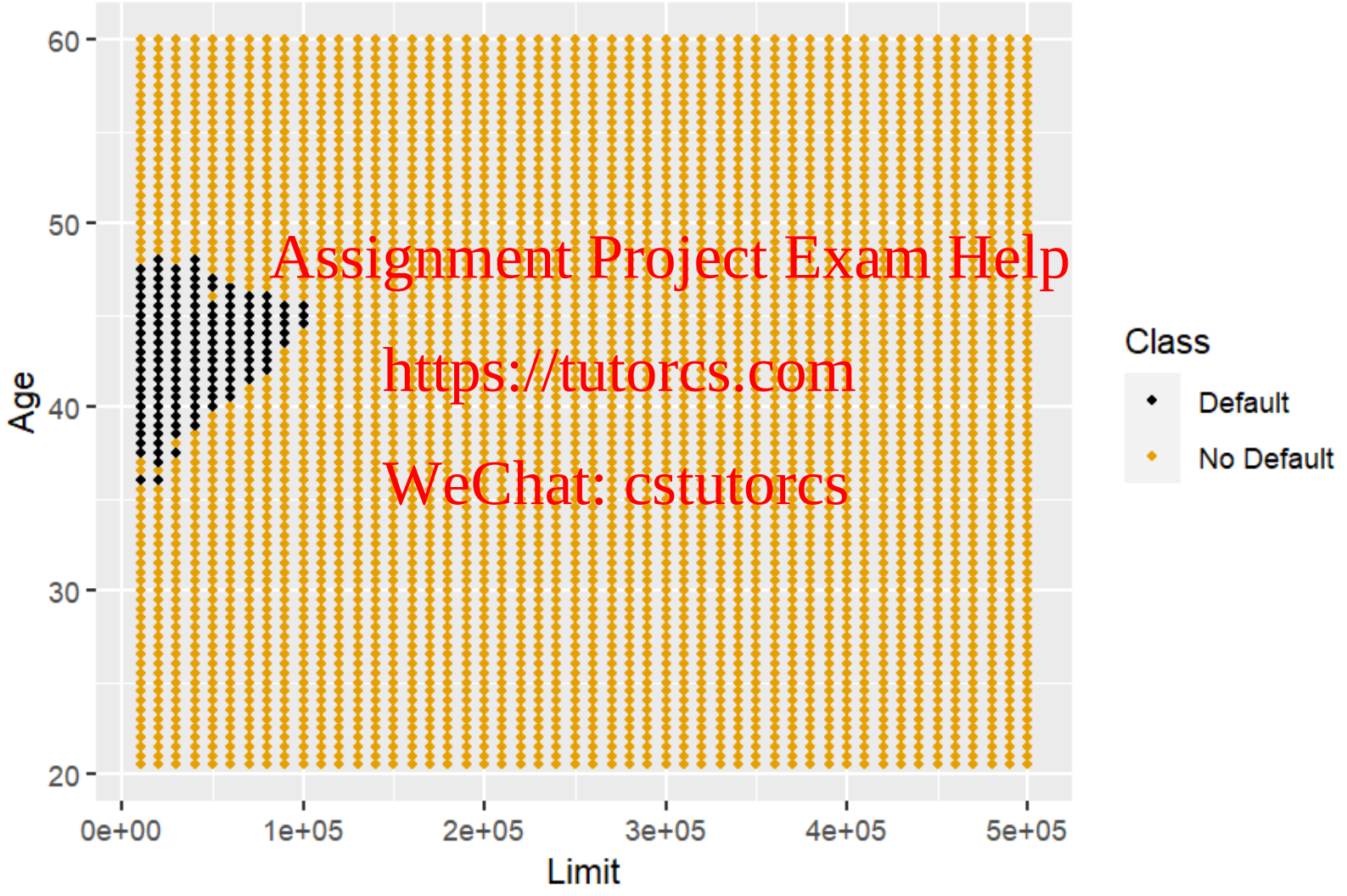
Decision boundary (k=1)



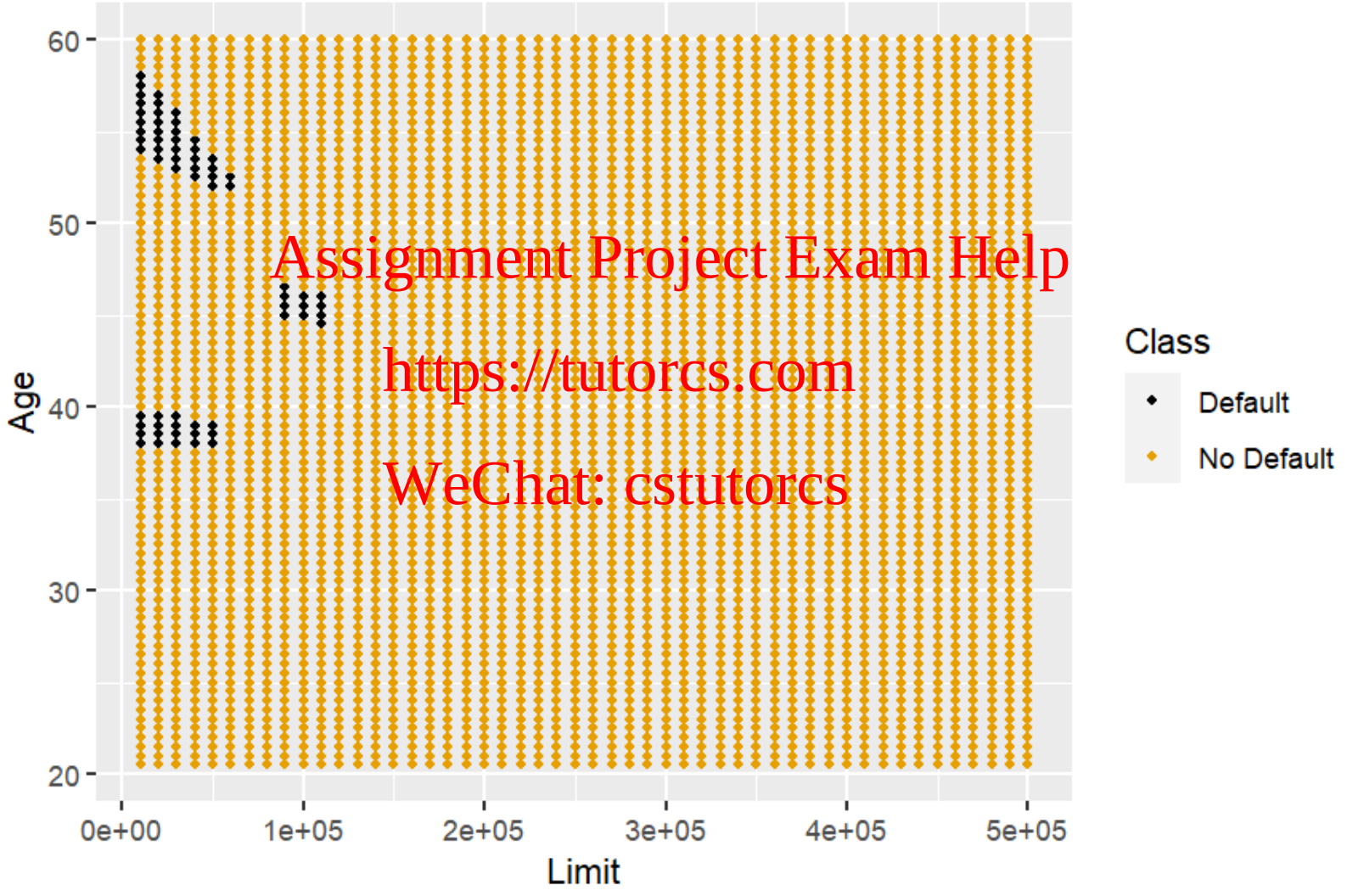
Decision boundary (k=3)



Decision boundary (k=11)



Decision boundary (k=17)



Choosing k

- Smaller values of k lead to very complicated decision boundaries.
- Larger values of k provide smoother boundaries although there are still some unstable regions.
- Larger values of k also slow down computation for big datasets.

Choosing k

- A general rule of thumb is to choose $k \approx \sqrt{n}$
- Also it is advised to choose a value of k that is NOT a multiple of the number of classes.
- This avoids ties.
- When ties are present, a class is chosen randomly.
- For the two class problem, choose odd k.

Choosing k

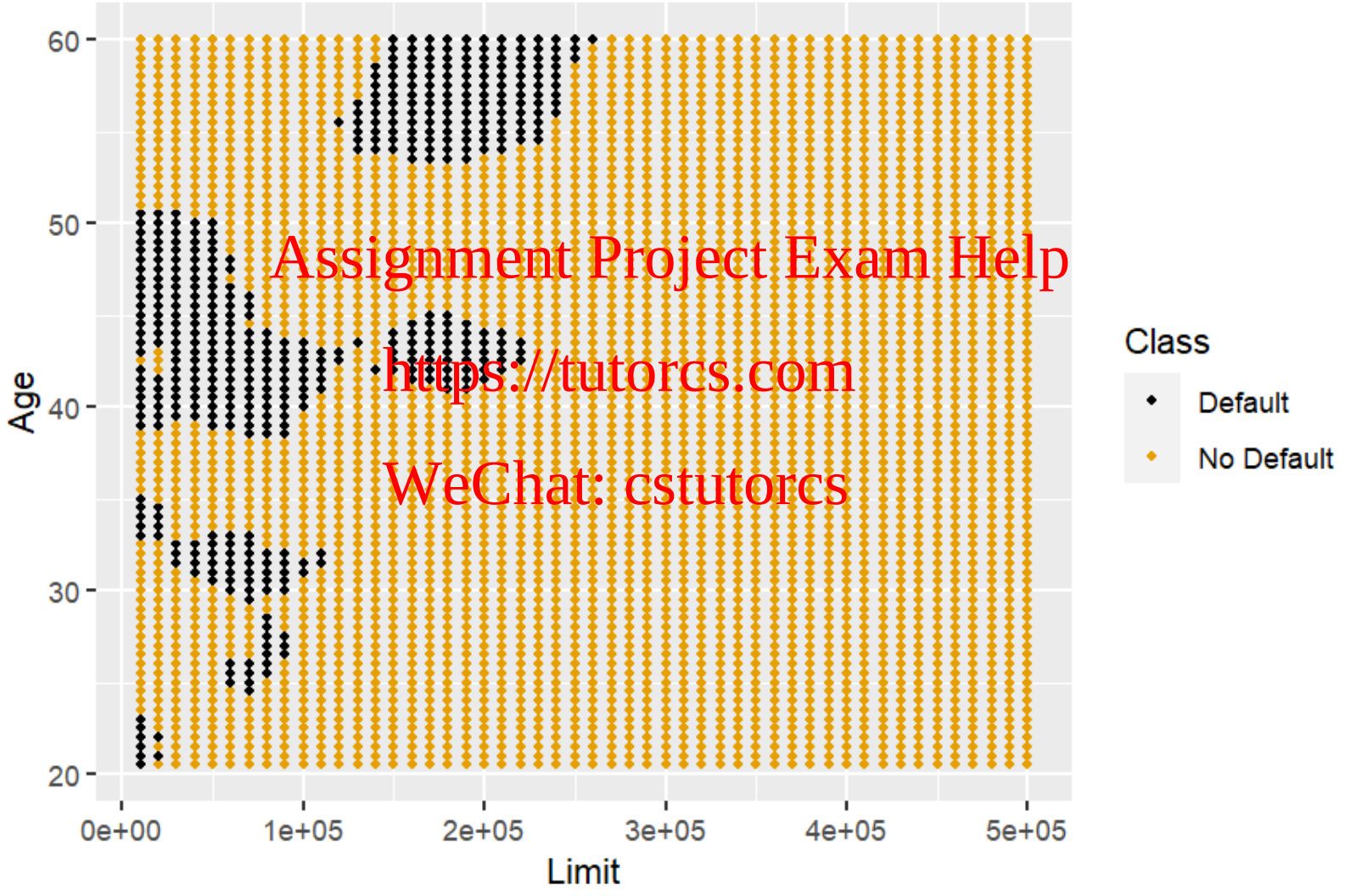
- Finally, data can be split into a training sample and test sample.
- We can then try many possible values of k.
- The optimal value can be selected so that the *test* misclassification error rate is minimised.
- Another alternative is to use cross validation to find the k that reduces the predicition error.

Instability of kNN

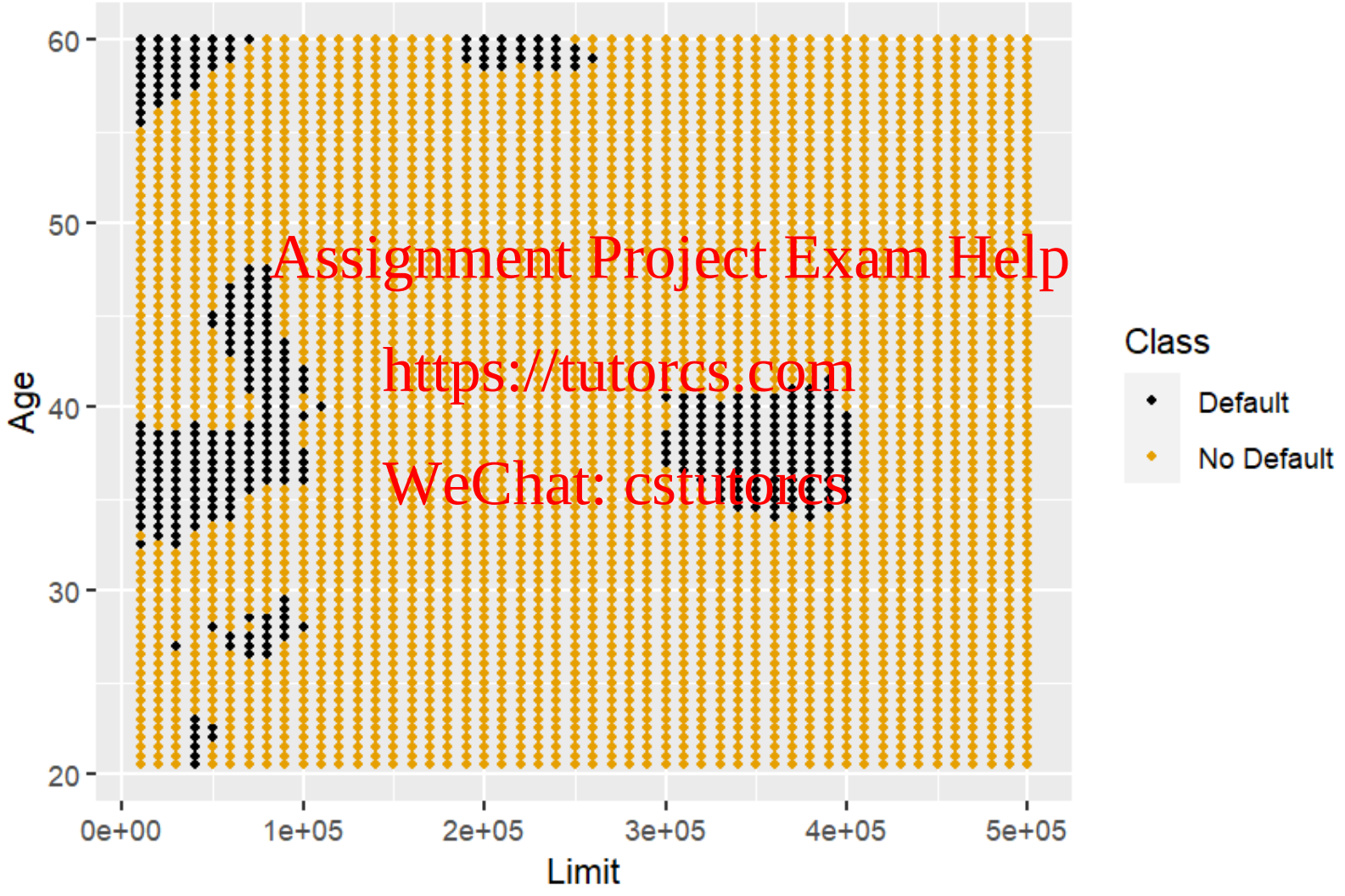
High variability

- A problem of k-NN classification is that it is highly variable.
- Now assume we use a fixed value of k but make small changes in the training data
- This change the decision boundaries dramatically.
- This is much more pronounced for smaller values of k.
- The next few slides show different subsamples of training data.

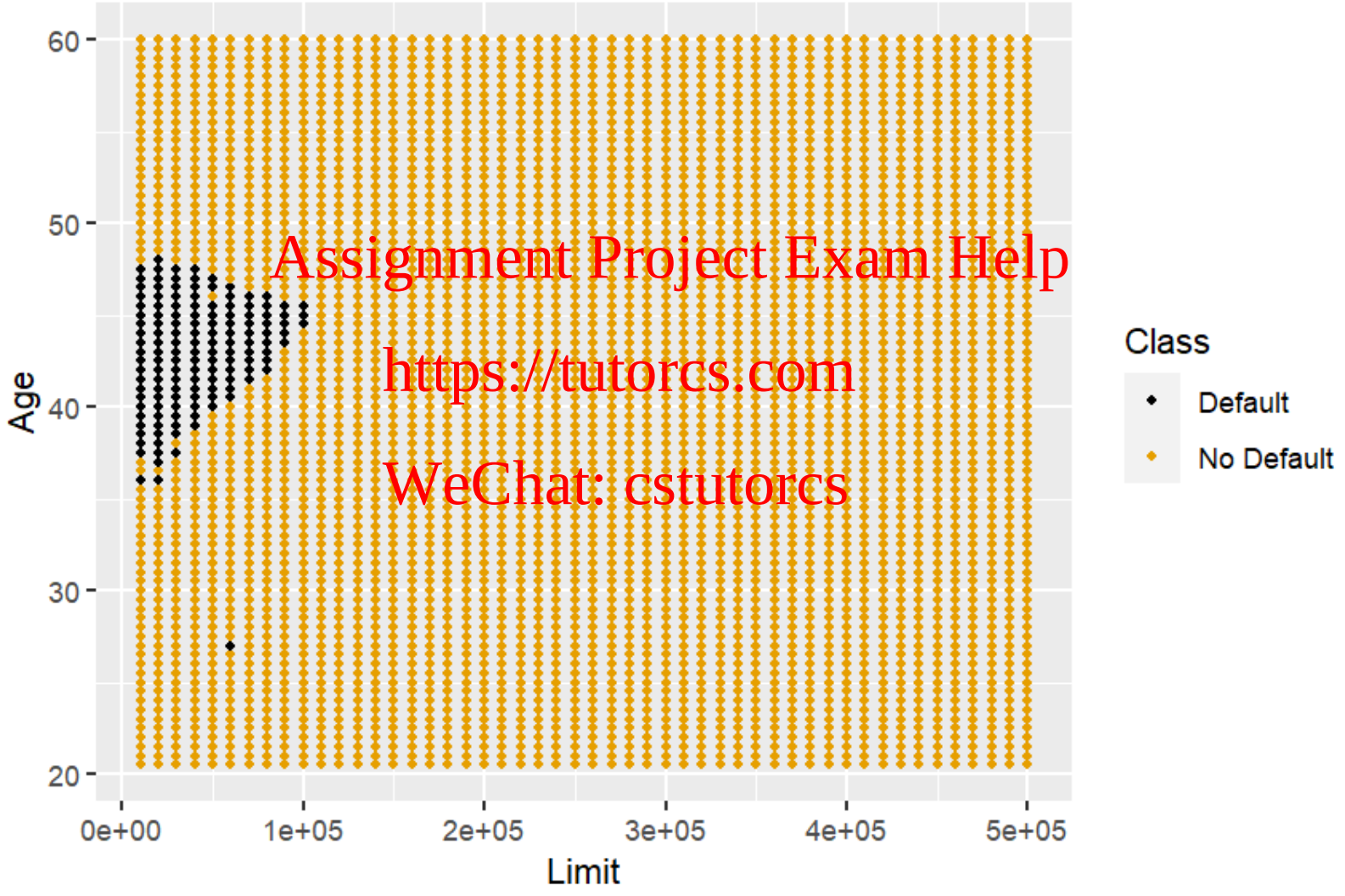
Original Training Sample (k=3)



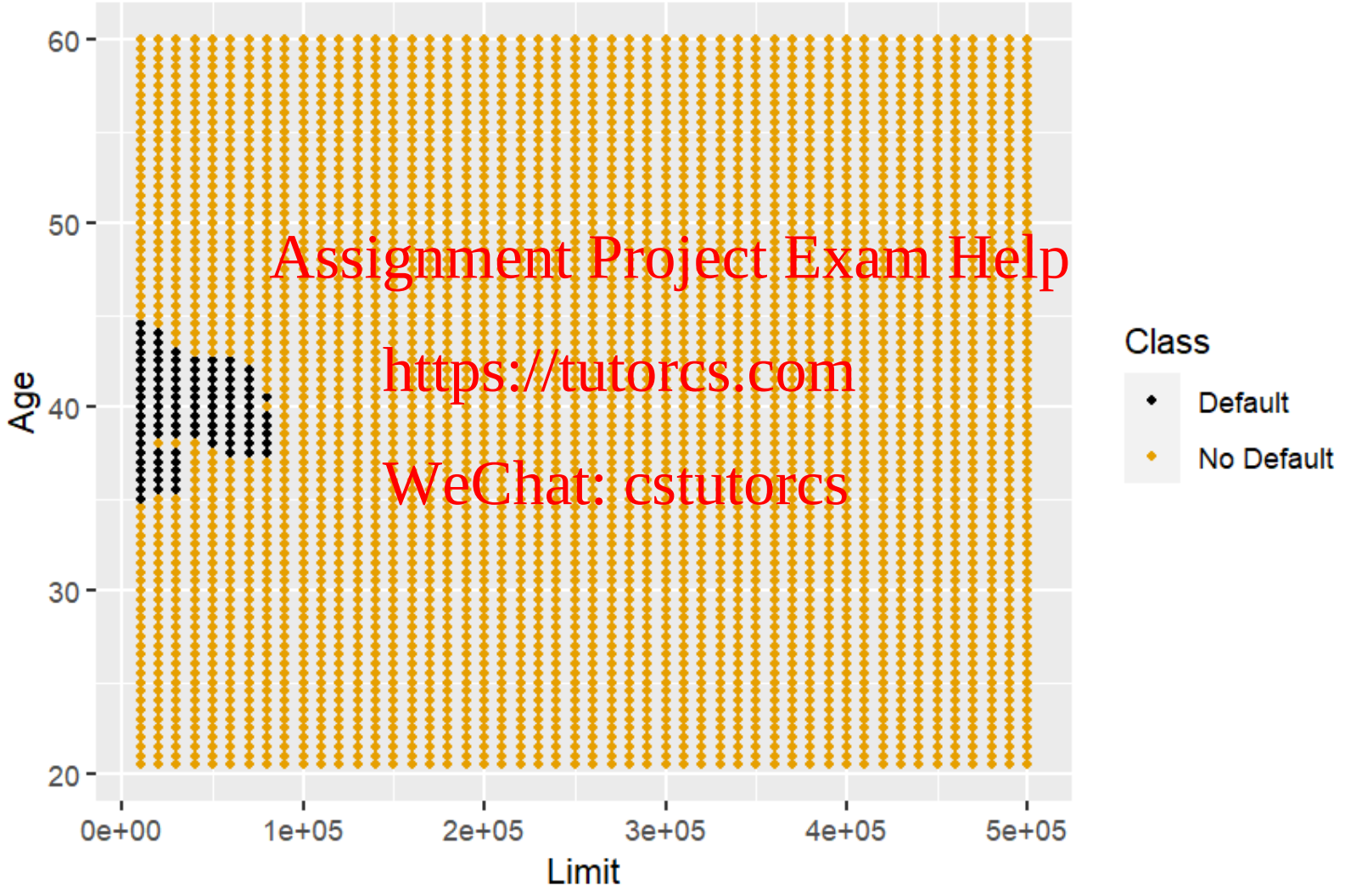
Different Training Sample (k=3)



Original Training Sample (k=11)



Different Training Sample (k=11)



An ethical problem

- A classifier with such high variability obviously poses statistical problems.
- It also poses ethical issues.
- Suppose the algorithm is used to decide who recieves a loan or who is audited by the tax office.
- It should be possible to clearly explain to this person why their loan application was rejected or why they were chosen for audit.
- Having such a sensitive algorithm makes this difficult.

k-NN Classification in R

Package

- The k-NN algorithm can be implemented using the `class` package.
- We will go through an example using the data that has already been cleaned and is in the file *knnexample.RData*
- Load this file using


```
load(here('data/knnexample.RData'))
```

Data

- A data frame of 250 randomly selected training observations for the predictors LIMIT_BAL and AGE (`train_x`)
- The values of the target variable for the training data stored as a factor (`train_y`)
- A data frame of 250 randomly selected test observations for the predictors LIMIT_BAL and AGE (`test_x`)
- The values of the target variable for the test data stored as a factor (`test_y`)

Standardisation

- An important issue with k-NN is that distances can be sensitive to units of measurement.
- For this reason we should standardise so that all predictors have a standard deviation of 1.

```
train_x_std<-scale(train_x)
mean_train_x<-attr(train_x_std,"scaled:center")
std_train_x<-attr(train_x_std,"scaled:scale")
```

Standardise training data

- The test data should also be standardised
- This tranformation should be identical for training and test data so using the mean and standard deviation of the training data.

```
test_x_std<-scale(test_x,
                  center = mean_train_x,
                  scale = std_train_x)
```


- Now the data are ready to be used in the knn function

Using knn

With k=1

```
yhat_k1 <- knn(train_x_std,test_x_std,
    train_y,k=1)
print(yhat_k1)
```

```
##   [1] No Default Default    Default    Default    No Default No Default
##   [7] No Default No Default No Default Default    No Default No Default
##  [13] No Default No Default No Default Default    No Default No Default
##  [19] No Default No Default Default    No Default Default    Default
##  [25] No Default No Default Default    No Default Default    No Default
##  [31] Default    No Default No Default No Default No Default No Default
##  [37] Default    No Default No Default Default    Default    No Default
##  [43] No Default No Default No Default No Default Default    No Default
##  [49] No Default No Default Default    No Default No Default No Default
##  [55] No Default No Default No Default Default    No Default No Default
##  [61] Default    No Default No Default Default    No Default No Default
##  [67] Default    Default    No Default No Default No Default No Default
```

Test misclassification rate

To compute test misclassification

```
miscl_k1<-mean(test_y!=yhat_k1)
print(miscl_k1)

## [1] 0.32
```


Overall, 32% of the test sample is incorrectly predicted

- As an exercise, you can now try to obtain the misclassification rate with $k = 5$

Answer

```
yhat_k5<- knn(train_x_std,
    test_x_std,
    train_y,k=5)
miscl_k5<-mean(test_y!=yhat_k5)
print(miscl_k5)
```

```
## [1] 0.292
```


The misclassification rate when $k = 5$ is 0.292.

Sensitivity and specificity

- It is also worth reporting results in a cross tabulation

```
table(test_y,yhat_k5)
```

```
##             yhat_k5
## test_y       Default No Default
##   Default          8         52
##   No Default      21        169
```


Letting "default" be the "positive" class, sensitivity is 8/(8+52) or 0.133, and specificity is 169/(21+169) or 0.889

Additional features

- If the argument `prob=TRUE` is included then the function returns probabilities.

```
prob_k5 <- knn(train_x_std,
    test_x_std,
    train_y,k=5,prob = TRUE)
```

Predictions

- The output still contains predictions

```
  print(prob_k5)
```

```
##   [1] No Default Default    No Default No Default Default    No Default
##   [7] No Default No Default No Default No Default No Default No Default
##  [13] No Default No Default No Default No Default No Default No Default
##  [19] No Default No Default No Default No Default No Default No Default
##  [25] No Default No Default No Default No Default Default    No Default
##  [31] No Default No Default No Default Default    Default    No Default
##  [37] No Default No Default No Default No Default No Default No Default
##  [43] No Default No Default No Default No Default No Default No Default
##  [49] No Default No Default No Default Default    Default    No Default
##  [55] No Default Default    Default    Default    No Default Default
##  [61] Default    No Default Default    Default    No Default No Default
##  [67] No Default No Default No Default No Default No Default No Default
##  [73] No Default No Default No Default No Default No Default No Default
##  [79] No Default No Default No Default No Default No Default No Default
```

Probabilities

- Probabilities can be stripped out using the `attr` function

```
  print(attr(prob_k5,"prob"))
```

```
##   [1] 1.0000000 0.5714286 0.6666667 0.6000000 0.6000000 0.8000000 1.0000000
##   [8] 1.0000000 0.6666667 1.0000000 0.6000000 0.8333333 1.0000000 0.5714286
##  [15] 1.0000000 0.6666667 1.0000000 1.0000000 0.6000000 1.0000000 0.6000000
##  [22] 0.8333333 0.6666667 0.6000000 1.0000000 1.0000000 0.5714286 1.0000000
##  [29] 0.6000000 0.5000000 0.6666667 0.8571429 1.0000000 0.6000000 0.6000000
##  [36] 0.8000000 0.6666667 0.6000000 1.0000000 0.5000000 0.6000000 0.6666667
##  [43] 0.6000000 0.6000000 0.8333333 1.0000000 0.6666667 0.8000000 1.0000000
##  [50] 1.0000000 0.8000000 0.6000000 0.6000000 1.0000000 1.0000000 0.6000000
##  [57] 0.6666667 0.8000000 1.0000000 0.6000000 0.8000000 1.0000000 0.6000000
##  [64] 0.6000000 0.6666667 0.8000000 0.5714286 0.5714286 1.0000000 1.0000000
##  [71] 0.6666667 0.6000000 1.0000000 0.6666667 0.8000000 0.8000000 1.0000000
##  [78] 0.8000000 0.8000000 1.0000000 0.6666667 0.6000000 1.0000000 0.8000000
##  [85] 0.5714286 1.0000000 0.8000000 0.6666667 1.0000000 0.8750000 1.0000000
##  [92] 1.0000000 0.8000000 0.7142857 0.6000000 0.7142857 1.0000000 0.5714429
```

Prediction and Probabilities

Prediction	Probability
No Default	1.0000000
Default	0.5714286
No Default	0.6666667
No Default	0.6000000

- For the first observation the probability of **No Default** is 1, but for the second observation the probablity of **Default** is 0.5714.
- The probability is relative to the prediction.

Ties in neighbours

- If there are only 5 neighbours all probabilities should be either 0.6, 0.8 or 1.
- This is not true. Why?
- There can be ties in finding the nearest neighbours.
- Two (or more) observations may tied as equally fifth closest.
- In this case both are used.

Regression

- The same principle can be used for regression.
- The predicted value is simply the average value of y for the k-nearest neighbours.
- This leads to a highly non-linear regression function.

Computational challenges

Where does the time to do this analysis go?

- Time to work with training data: negligible (standardization)
- Time to make predictions: Extensive (need to work out the distance between each observation we want to predict and every other training observation)
- This is called being a "lazy learner" (when learning is delayed until it needs to make a prediction) and can impact implementation in real time applications.
- Lazy learning is useful when real time updates are desirable.
- One solution is to dimension reduce (remember our clustering lectures?)
- Also have computational issues with lots of predictors (n)
- This is known as the curse of dimensionality

Conclusion

- Using k-NN is a simple but often effective method for classification.
- There are limitations
  - Complicated decision boundaries.
  - High variance.
  - Also it works poorly when there is a large number of predictors.

MONASH University

Lecturer: *Lauren Kennedy*

Department of Econometrics and Business Statistics

lauren.kennedy1@monash.edu

Week 11