
multiverse

multiverse

Data Science Unit 3

Logistic Regression

Before we start…

- ➔ Make sure you are comfortable
- ➔ Have water and maybe a strong coffee handy
- ➔ If you need a break… take it!
- ➔ If you need a stretch – please go ahead!
- ➔ Please mute yourselves if you are not talking
- ➔ Have your video on at all times

…and let's get started!

Session Outline…

Day 1

- Intro
- Data Life Cycle
- Data Analytics Life Cycle
- Communication and Dealing with Stakeholders
- Project Briefs

Day 2

- Data Types
- Data Sources
- Data Formats
- Extract/Transform/Load (ETL)
- Wrap Up and Assignment

...and then you are done!

In this session we will...

- **Define** logistic regression
- **Explain** how logistic regression works
- **Build** a logistic regression model with categorical labels
- **Interpret** coefficients from logistic regression
- **Compute** performance metrics for classification models
- **Construct** a confusion matrix to help us evaluate classification models

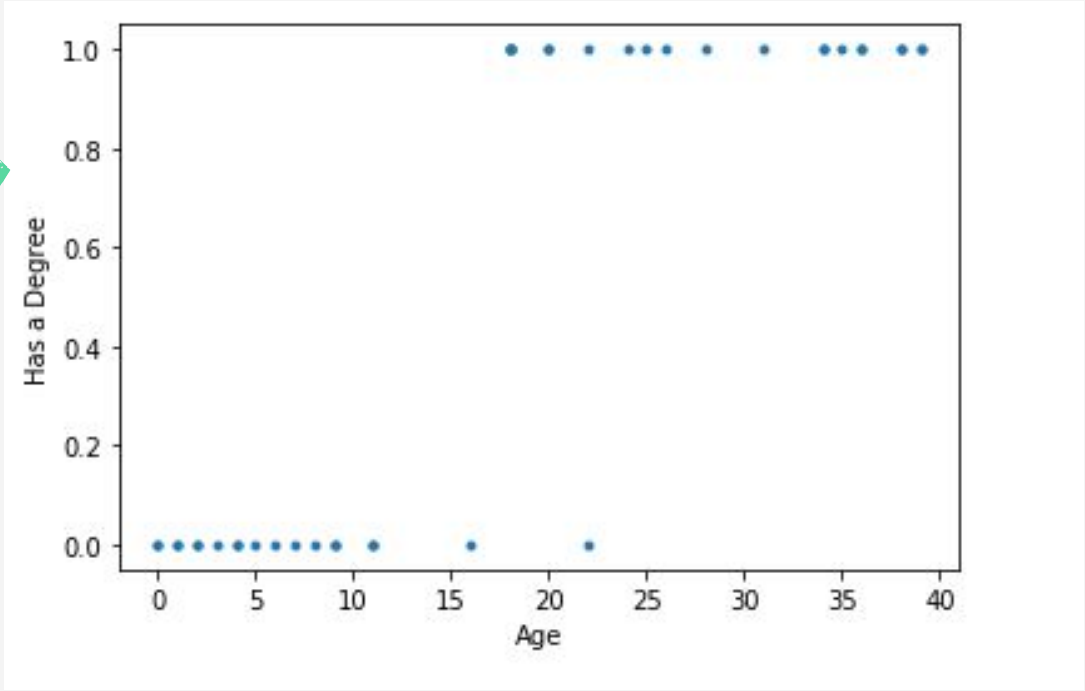
multiverse

Linear Regression Warm Up

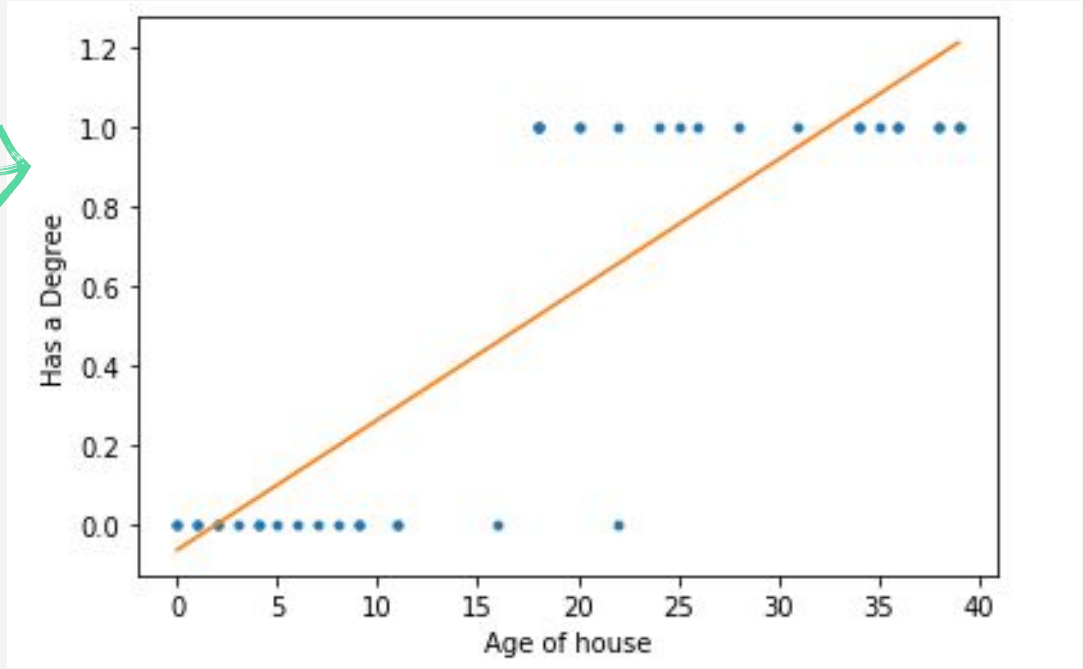
multiverse

Logistic Regression

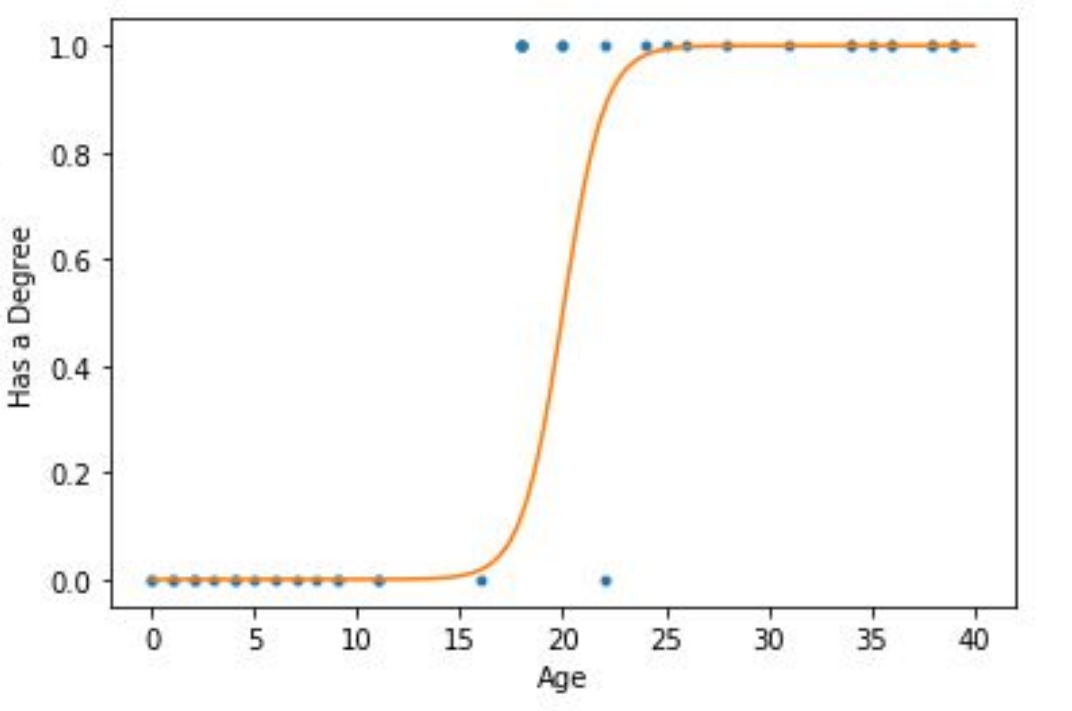
Logistic Regression



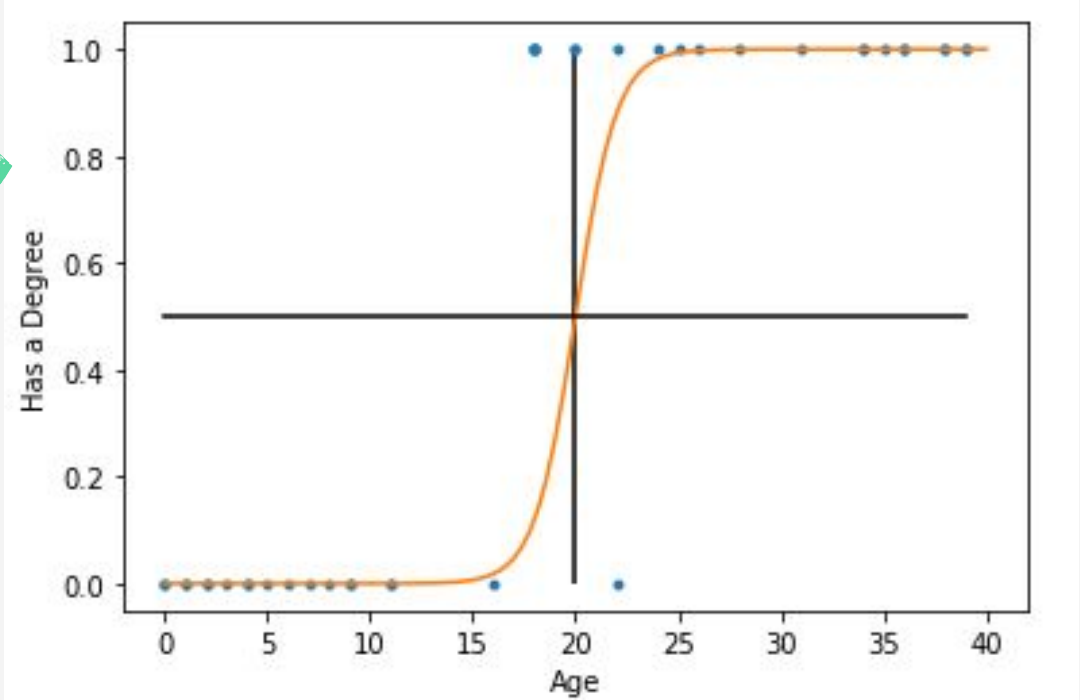
Logistic Regression



Logistic Regression



Logistic Regression



Building a Model

```python
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()

logreg.fit(X,y)
```

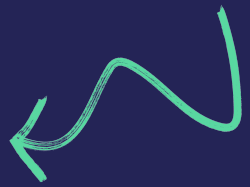
```python
logreg.predict(X)
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

Building a Model

```
logreg.predict_proba(X)
```

```
array([[0.9939759 , 0.0060241 ],
       [0.99296771, 0.00703229],
       [0.98949363, 0.01050637],
       [0.98949363, 0.01050637],
       [0.98811597, 0.01188403],
       [0.98614074, 0.01385926],
       [0.98614074, 0.01385926],
       [0.98526292, 0.01473708],
       [0.9817376 , 0.0182624 ],
       [0.98117062, 0.01882938],
       [0.98058638, 0.01941362],
       [0.97806663, 0.02193337],
       [0.97738831, 0.02261169],
       [0.9752281 , 0.0247719 ],
       [0.9752281 , 0.0247719 ],
       [0.97367755, 0.02632245],
```



multiverse

Let’s Practice

multiverse

Comparison to other Models

Comparison

Advantages of logistic regression:

- ➔ Highly interpretable (if you remember how).
- ➔ Model training and prediction are fast.
- ➔ No tuning is required (excluding regularisation).
- ➔ Features don't need scaling.
- ➔ Can perform well with a small number of observations.
- ➔ Outputs well-calibrated predicted probabilities.

Disadvantages of logistic regression:

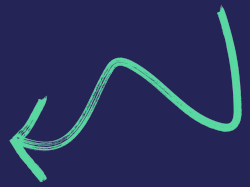
- ➔ Presumes a linear relationship between the features.
- ➔ Performance is (generally) not competitive with the best supervised learning methods.
- ➔ Can't automatically learn feature interactions.

multiverse

Evaluating our model

Accuracy

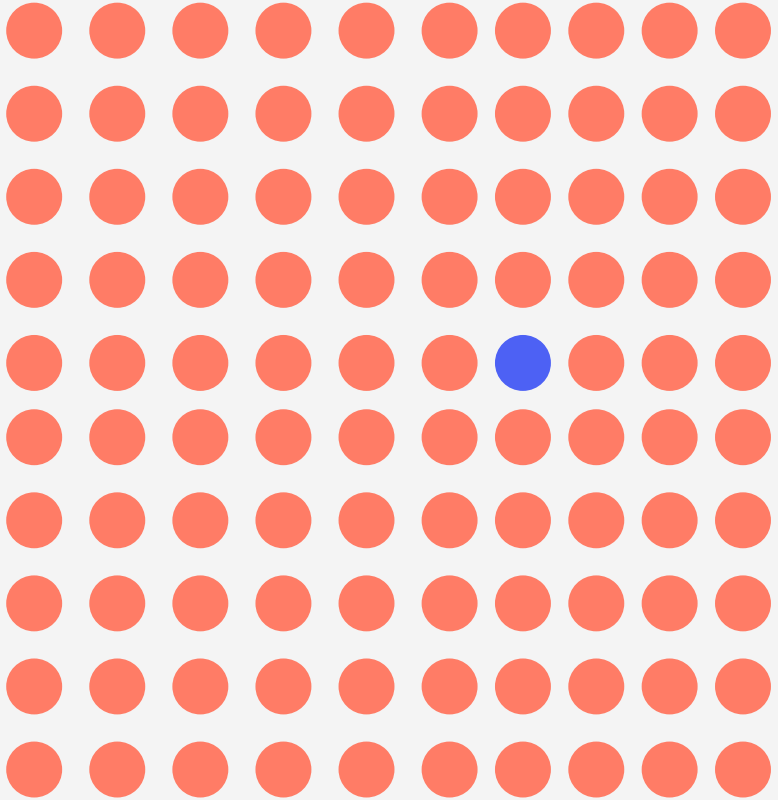
$$Accuracy = \frac{total\ predicted\ correct}{total\ predicted}$$



```
lr.score(X_test,y_test)
```

```
0.8769230769230769
```

Baseline



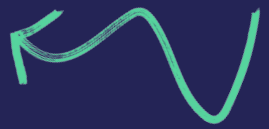
Baseline

```
lr.score(X_test,y_test)
```

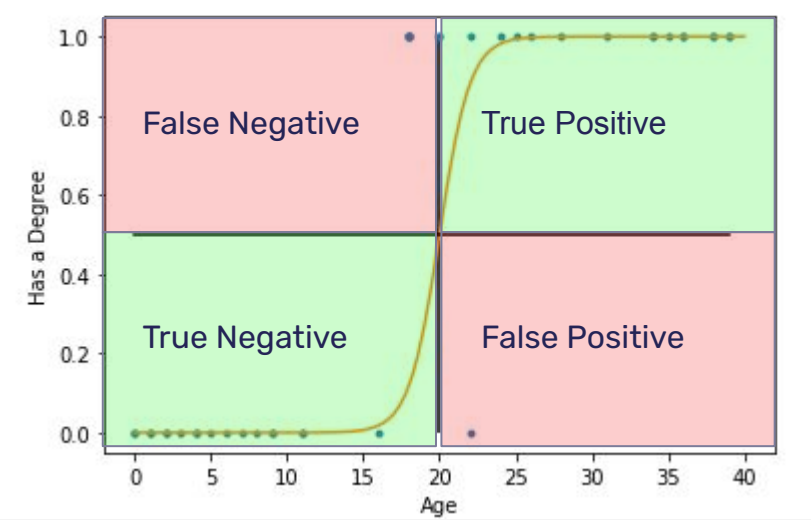
```
0.8769230769230769
```

```
glass['household'].value_counts(normalize=True).max()
```

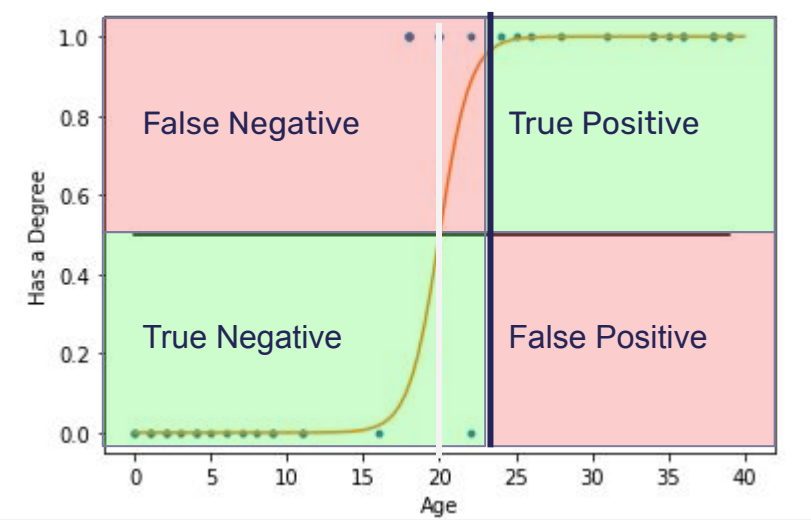
```
0.7616822429906542
```



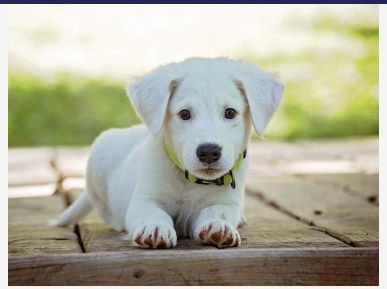
Confusion!



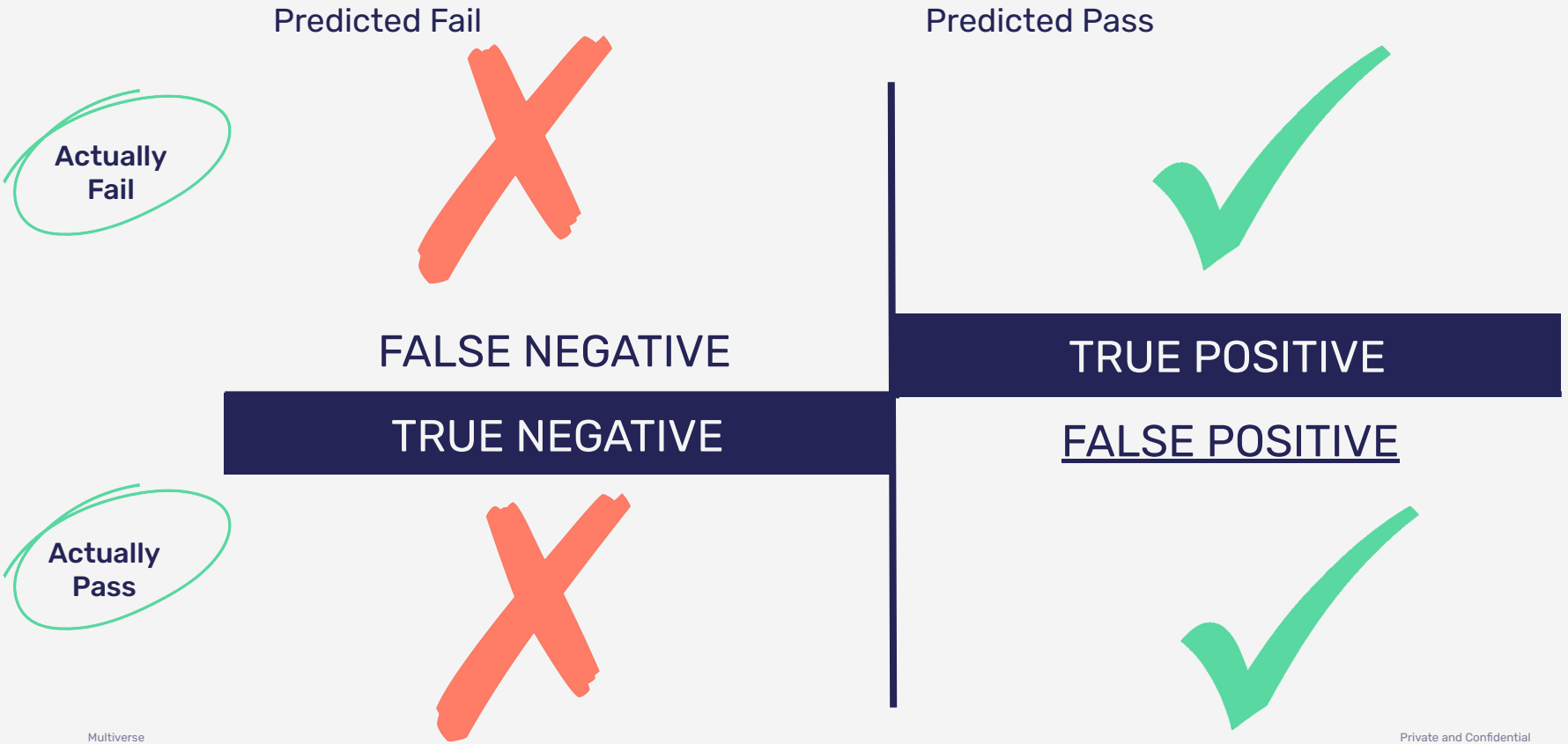
Confusion!



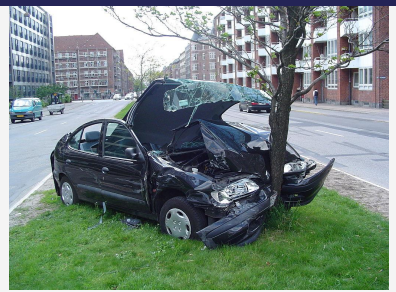
Confusion!

	Predicted Dog	Predicted Cat
Actually Cat	 FALSE NEGATIVE	 TRUE POSITIVE
Actually Dog	TRUE NEGATIVE 	FALSE POSITIVE 

Confusion!



Confusion!

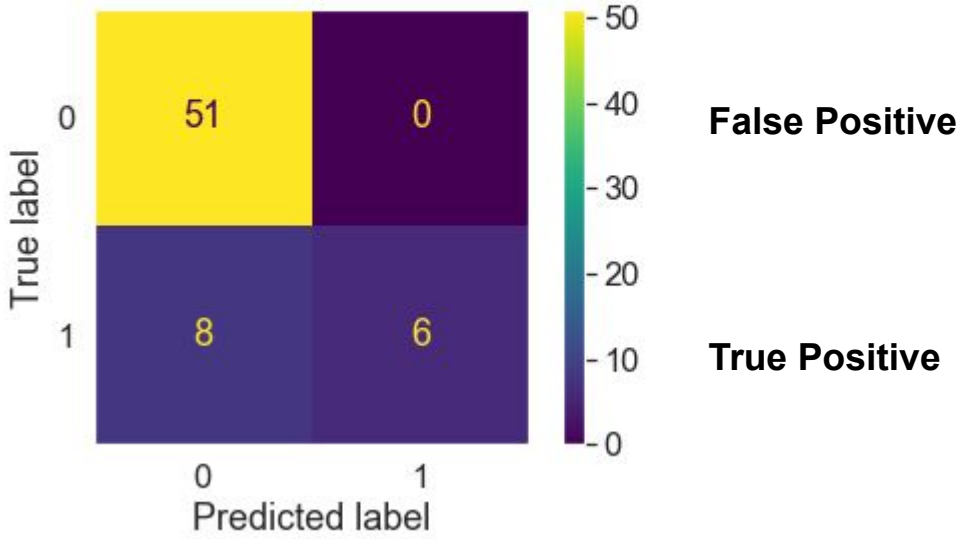
	Predicted Failed MOT	Predicted Passed MOT
Actually Passed MOT	 FALSE NEGATIVE	 TRUE POSITIVE
Actually Failed MOT	TRUE NEGATIVE 	FALSE POSITIVE 

Confusion!

True Negative

False Negative

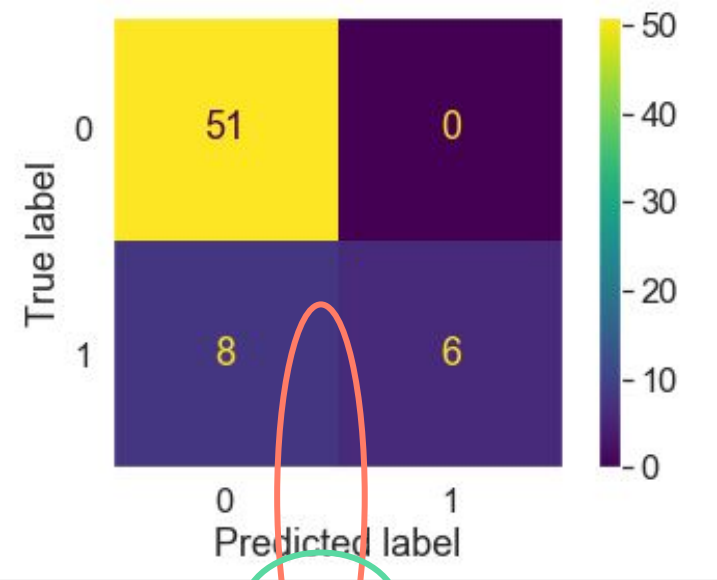
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(lr,X_test,y_test)
plt.grid(False);
```



False Positive

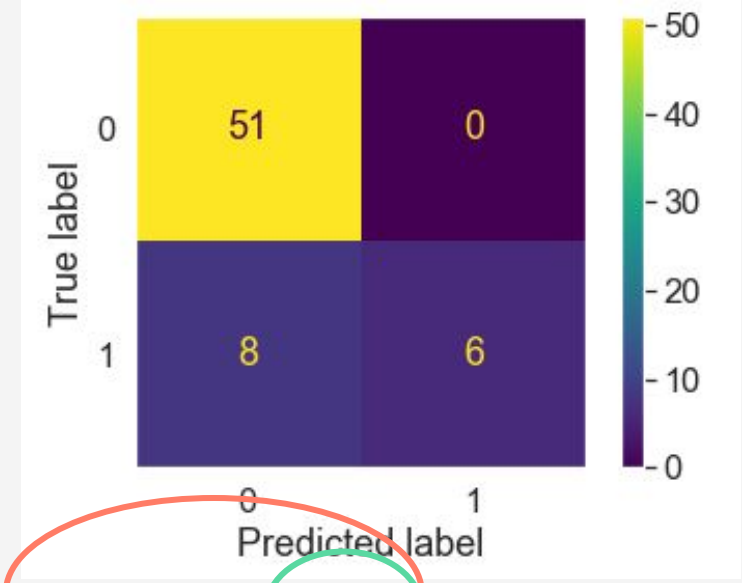
True Positive

Confusion!



$$precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Confusion!



$$recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Confusion!

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import precision_score, recall_score, classification_report
```

```
y_pred=lr.predict(X_test)
```

```
print('Recall: ' + str(recall_score(y_test,y_pred)))
print('Precision: ' + str(precision_score(y_test,y_pred)))
```

```
Recall: 0.42857142857142855
Precision: 1.0
```

Confusion!

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.86	1.00	0.93	51
1	1.00	0.43	0.60	14
accuracy			0.88	65
macro avg	0.93	0.71	0.76	65
weighted avg	0.89	0.88	0.86	65

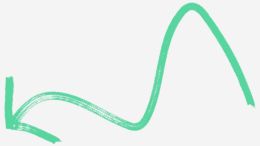
multiverse

Let’s Practice

multiverse

Summary

Summary



Logistic Regression

- ➔ What kind of machine learning problems does logistical regression address?
- ➔ What do the coefficients in logistic regression represent? How does the interpretation differ from ordinary least squares? How is it similar

The confusion matrix

- ➔ How do true positive rate and false positive rate help explain accuracy?
- ➔ Why might one classification metric be more import to tune than another? Give an example of business problem or project where this would be the case.

multiverse

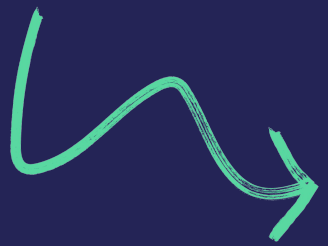
Practice

multiverse

Quiz Time!

multiverse

Thank you



Get in touch
info@multiverse.io