

# Logistic regression for probability of default

CREDIT RISK MODELING IN PYTHON



**Michael Crabtree**

Data Scientist, Ford Motor Company

# Probability of default

- The likelihood that someone will default on a loan is the probability of default
- A probability value between 0 and 1 like 0.86
- `loan_status` of 1 is a default or 0 for non-default

# Probability of default

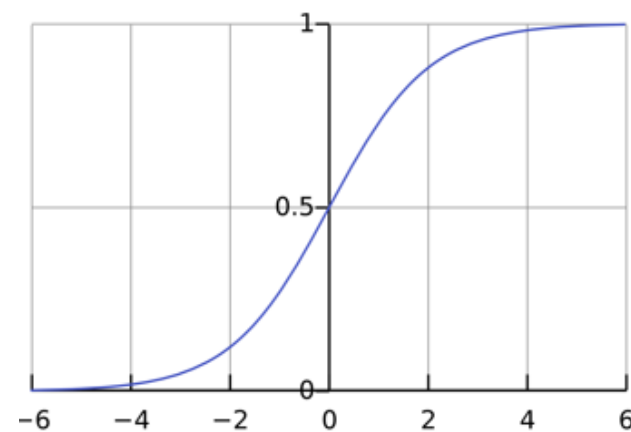
- The likelihood that someone will default on a loan is the probability of default
- A probability value between 0 and 1 like 0.86
- `loan_status` of 1 is a default or 0 for non-default

Probability of Default	Interpretation	Predicted loan status
0.4	Unlikely to default	0
0.90	Very likely to default	1
0.1	Very unlikely to default	0

# Predicting probabilities

- Probabilities of default as an outcome from machine learning
  - Learn from data in columns (features)
- Classification models (default, non-default)
- Two most common models:
  - Logistic regression
  - Decision tree

Logistic Regression



Decision Tree

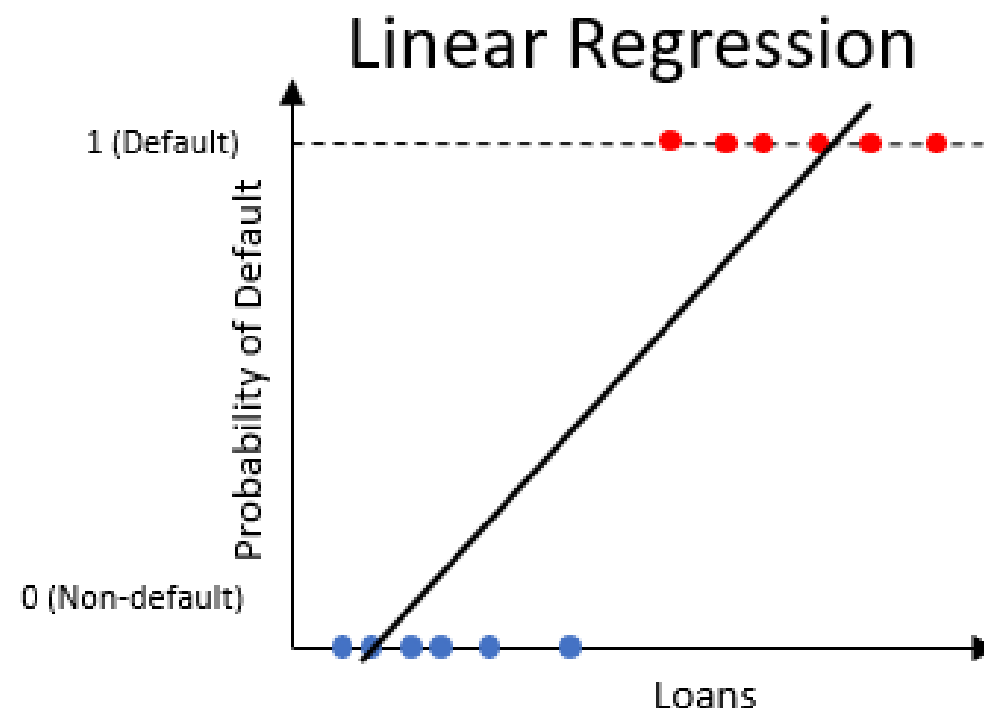


# Logistic regression

- Similar to the linear regression, but only produces values between 0 and 1

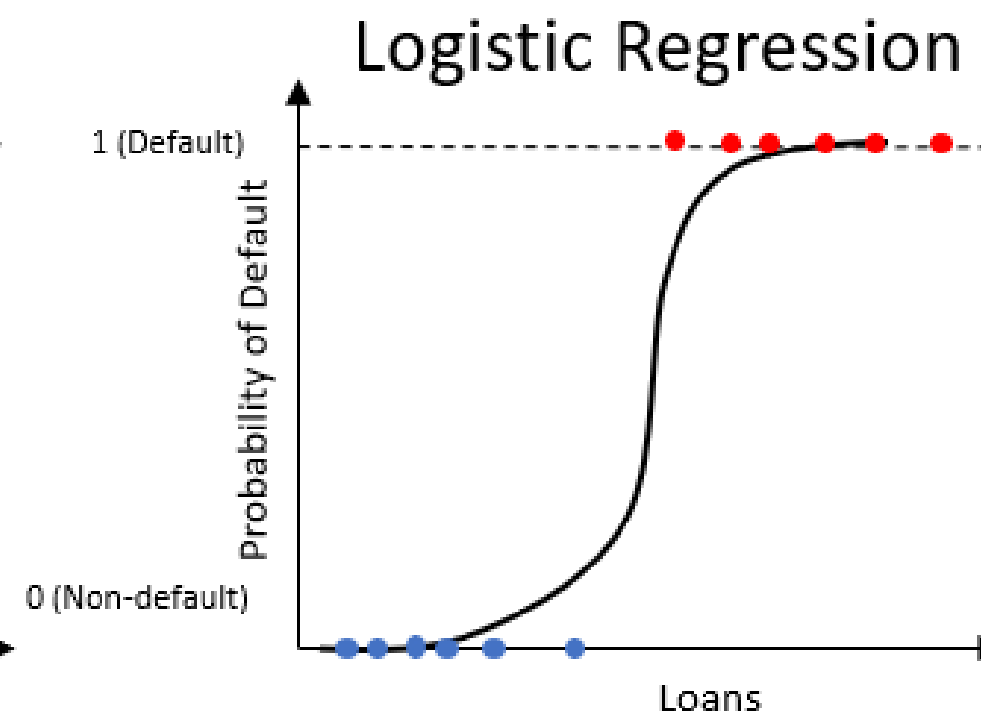
## Linear Regression

$$Y = \beta_0 + (\beta_1 * X_1) + (\beta_2 * X_2) \dots$$



## Logistic Regression

$$P(\text{loan\_status} = 1) = \frac{1}{1 + e^{-Y}}$$



# Training a logistic regression

- Logistic regression available within the scikit-learn package

```
from sklearn.linear_model import LogisticRegression
```

- Called as a function with or without parameters

```
clf_logistic = LogisticRegression(solver='lbfgs')
```

- Uses the method `.fit()` to train

```
clf_logistic.fit(training_columns, np.ravel(training_labels))
```

- Training Columns: all of the columns in our data **except** `loan_status`
- Labels: `loan_status` (0,1)

# Training and testing

- Entire data set is usually split into two parts

# Training and testing

- Entire data set is usually split into two parts

Data Subset	Usage	Portion
Train	Learn from the data to generate predictions	60%
Test	Test learning on new unseen data	40%



# Creating the training and test sets

- Separate the data into training columns and labels

```
X = cr_loan.drop('loan_status', axis = 1)
y = cr_loan[['loan_status']]
```

- Use `train_test_split()` function already within sci-kit learn

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.4, random_state=123)
```

- `test_size` : percentage of data for test set
- `random_state` : a random seed value for reproducibility

# Let's practice!

CREDIT RISK MODELING IN PYTHON

# Predicting the probability of default

CREDIT RISK MODELING IN PYTHON



**Michael Crabtree**

Data Scientist, Ford Motor Company

# Logistic regression coefficients

```
# Model Intercept  
array([-3.30582292e-10])  
# Coefficients for ['loan_int_rate', 'person_emp_length', 'person_income']  
array([[ 1.28517496e-09, -2.27622202e-09, -2.17211991e-05]])
```

$$P(\text{loan\_status} = 1) = \frac{1}{1 + e^{-(-3.3e^{-10} + (1.29e^{-9} * \text{Int Rate}) + (-2.28e^{-9} * \text{Emp Length}) + (-2.17e^{-5} * \text{Income}))}}$$

```
# Calculating probability of default  
int_coef_sum = -3.3e-10 +  
    (1.29e-09 * loan_int_rate) + (-2.28e-09 * person_emp_length) + (-2.17e-05 * person_income)  
prob_default = 1 / (1 + np.exp(-int_coef_sum))  
prob_nondefault = 1 - (1 / (1 + np.exp(-int_coef_sum)))
```

# Interpreting coefficients

```
# Intercept  
intercept = -1.02  
# Coefficient for employment length  
person_emp_length_coef = -0.056
```

- For every 1 year increase in `person_emp_length`, the person is less likely to default

# Interpreting coefficients

```
# Intercept
intercept = -1.02
# Coefficient for employment length
person_emp_length_coef = -0.056
```

- For every 1 year increase in `person_emp_length`, the person is less likely to default

intercept	person_emp_length	value * coef	probability of default
-1.02	10	$(10 * -0.06)$	.17
-1.02	11	$(11 * -0.06)$	.16
-1.02	12	$(12 * -0.06)$	.15

# Using non-numeric columns

- Numeric: `loan_int_rate` , `person_emp_length` , `person_income`
- Non-numeric:

```
cr_loan_clean['loan_intent']
```

```
EDUCATION  
MEDICAL  
VENTURE  
PERSONAL  
DEBTCONSOLIDATION  
HOMEIMPROVEMENT
```

- Will cause errors with machine learning models in Python unless processed

# One-hot encoding

- Represent a string with a number

	person_age	person_income	loan_intent
0	21	9600	EDUCATION
1	25	9600	MEDICAL
2	23	65500	MEDICAL
3	24	54400	MEDICAL
4	21	9900	VENTURE



# One-hot encoding

- Represent a string with a number
- `0` or `1` in a new column `column_VALUE`

	person_age	person_income	loan_intent
0	21	9600	EDUCATION
1	25	9600	MEDICAL
2	23	65500	MEDICAL
3	24	54400	MEDICAL
4	21	9900	VENTURE

	person_age	person_income	loan_intent_EDUCATION	loan_intent_MEDICAL	loan_intent_VENTURE
0	21	9600	1	0	0
1	25	9600	0	1	0
2	23	65500	0	1	0
3	24	54400	0	1	0
4	21	9900	0	0	1

# Get dummies

- Utilize the `get_dummies()` within `pandas`

```
# Separate the numeric columns
cred_num = cr_loan.select_dtypes(exclude=['object'])
# Separate non-numeric columns
cred_cat = cr_loan.select_dtypes(include=['object'])
# One-hot encode the non-numeric columns only
cred_cat_onehot = pd.get_dummies(cred_cat)
# Union the numeric columns with the one-hot encoded columns
cr_loan = pd.concat([cred_num, cred_cat_onehot], axis=1)
```

# Predicting the future, probably

- Use the `.predict_proba()` method within scikit-learn

```
# Train the model
clf_logistic.fit(X_train, np.ravel(y_train))
# Predict using the model
clf_logistic.predict_proba(X_test)
```

- Creates array of probabilities of default

```
# Probabilities: [[non-default, default]]
array([[0.55, 0.45]])
```

# Let's practice!

CREDIT RISK MODELING IN PYTHON

# Credit model performance

CREDIT RISK MODELING IN PYTHON



**Michael Crabtree**

Data Scientist, Ford Motor Company

# Model accuracy scoring

- Calculate accuracy

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of predictions}}$$

- Use the `.score()` method from scikit-learn

```
# Check the accuracy against the test data  
clf_logistic1.score(X_test, y_test)
```

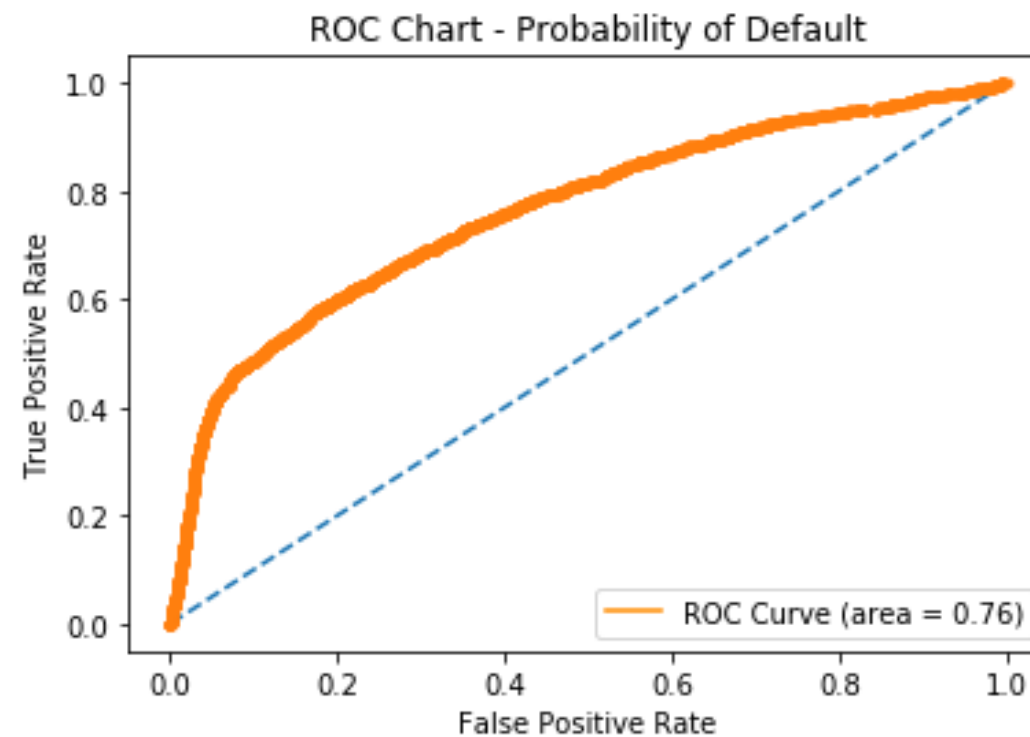
```
0.81
```

- 81% of values for `loan_status` predicted correctly

# ROC curve charts

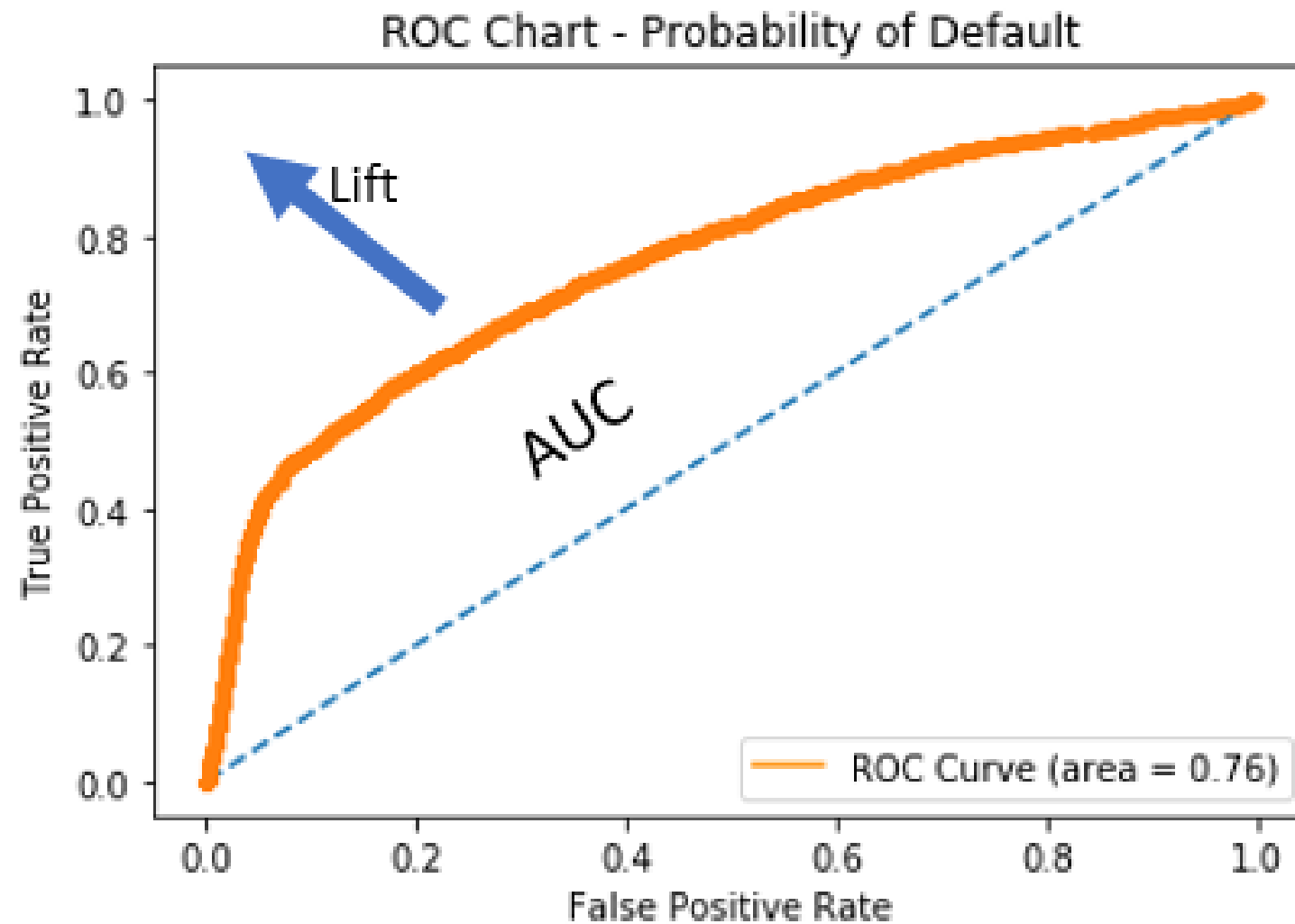
- Receiver Operating Characteristic curve
  - Plots true positive rate (sensitivity) against false positive rate (fall-out)

```
fallout, sensitivity, thresholds = roc_curve(y_test, prob_default)
plt.plot(fallout, sensitivity, color = 'darkorange')
```



# Analyzing ROC charts

- Area Under Curve (AUC): area between curve and random prediction

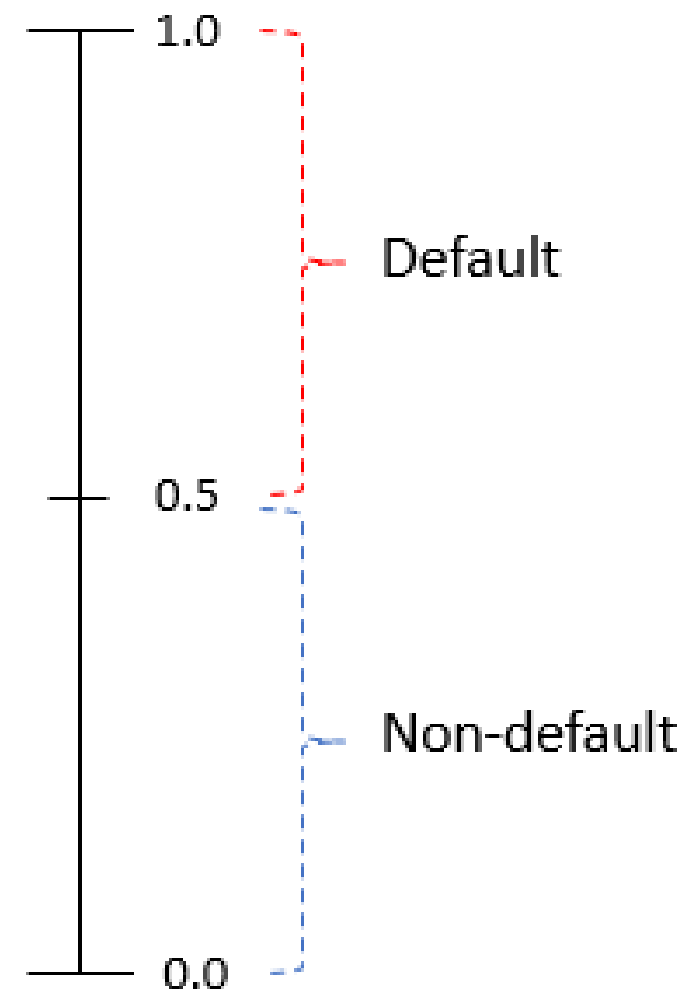




# Default thresholds

- Threshold: at what point a probability is a default

Predicted probability



Predicted probability	New loan_status
0.45	0 (non-default)
0.87	1 (default)

# Setting the threshold

- Relabel loans based on our threshold of 0.5

```
preds = clf_logistic.predict_proba(X_test)
preds_df = pd.DataFrame(preds[:,1], columns = ['prob_default'])
preds_df['loan_status'] = preds_df['prob_default'].apply(lambda x: 1 if x > 0.5 else 0)
```

	prob_default	loan_status
11779	0.079626	0
11780	0.051979	0
11781	0.522450	1
11782	0.370478	0
11783	0.123786	0

# Credit classification reports

- `classification_report()` within scikit-learn

```
from sklearn.metrics import classification_report
classification_report(y_test, preds_df['loan_status'], target_names=target_names)
```

	precision	recall	f1-score	support
Non-Default	0.79	0.99	0.88	9198
Default	0.67	0.04	0.07	2586
micro avg	0.78	0.78	0.78	11784
macro avg	0.73	0.52	0.47	11784
weighted avg	0.76	0.78	0.70	11784

# Selecting classification metrics

- Select and store specific components from the `classification_report()`
- Use the `precision_recall_fscore_support()` function from scikit-learn

	precision	recall	f1-score	support
Non-Default	0.79	0.99	0.88	9198
Default	0.67	0.04	0.07	2586
micro avg	0.78	0.78	0.78	11784
macro avg	0.73	0.52	0.47	11784
weighted avg	0.76	0.78	0.70	11784

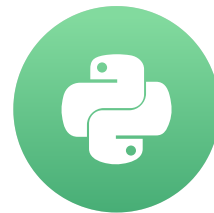
```
from sklearn.metrics import precision_recall_fscore_support
precision_recall_fscore_support(y_test, preds_df['loan_status'])[1][1]
```

# Let's practice!

CREDIT RISK MODELING IN PYTHON

# Model discrimination and impact

CREDIT RISK MODELING IN PYTHON



**Michael Crabtree**

Data Scientist, Ford Motor Company

# Confusion matrices

- Shows the number of correct and incorrect predictions for each `loan_status`

<b>True Negatives</b> (Predicted = 0, Actual = 0)	<b>False Positives</b> (Predicted = 1, Actual = 0)
<b>False Negatives</b> (Predicted = 0, Actual = 1)	<b>True Positives</b> (Predicted = 1, Actual = 1)

$$Precision(0) = \frac{TN}{TN + FN}$$

$$Precision(1) = \frac{TP}{TP + FP}$$

$$Recall(0) = \frac{TN}{TN + FP} \quad Recall(1) = \frac{TP}{TP + FN}$$

# Default recall for loan status

- Default recall (or sensitivity) is the proportion of true defaults predicted

	precision	recall	f1-score	support
Non-Default	0.79	0.99	0.88	9198
Default	0.67	0.04	0.07	2586
micro avg	0.78	0.78	0.78	11784
macro avg	0.73	0.52	0.47	11784
weighted avg	0.76	0.78	0.78	11784

$$Recall(1) = \frac{TP}{TP + FN}$$



# Recall portfolio impact

- Classification report - Underperforming Logistic Regression model

	precision	recall	f1-score	support
Non-Default	0.79	0.99	0.88	9198
Default	0.67	0.04	0.07	2586
micro avg	0.78	0.78	0.78	11784
macro avg	0.73	0.52	0.47	11784
weighted avg	0.76	0.78	0.70	11784

# Recall portfolio impact

- Classification report - Underperforming Logistic Regression model

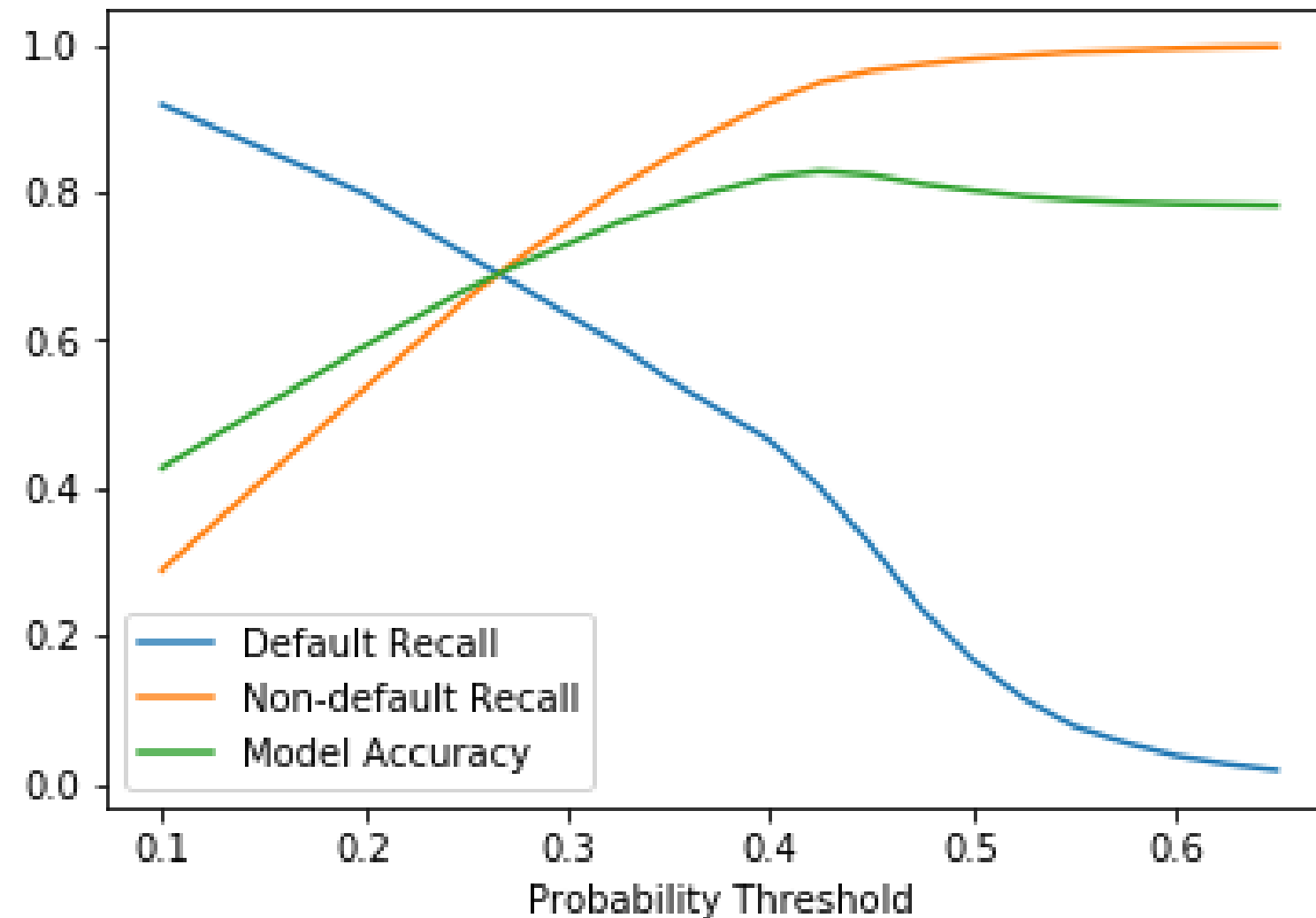
	precision	recall	f1-score	support
Non-Default	0.79	0.99	0.88	9198
Default	0.67	0.04	0.07	2586
micro avg	0.78	0.78	0.78	11784
macro avg	0.73	0.52	0.47	11784
weighted avg	0.76	0.78	0.70	11784

- Number of true defaults: 50,000

Loan Amount	Defaults Predicted / Not Predicted	Estimated Loss on Defaults
\$50	.04 / .96	$(50000 \times .96) \times 50 = \$2,400,000$

# Recall, precision, and accuracy

- Difficult to maximize all of them because there is a trade-off



# Let's practice!

CREDIT RISK MODELING IN PYTHON