Model Evaluation and Selection

Part 2

Classifier Decision Functions

Decision Functions (decision_function)

- Each classifier score value per test point indicates how confidently the classifier predicts the positive class (large-magnitude positive values) or the negative class (large-magnitude negative values).
- Choosing a fixed decision threshold gives a classification rule.
- By sweeping the decision threshold through the entire range of possible score values, we get a series of classification outcomes that form a curve.

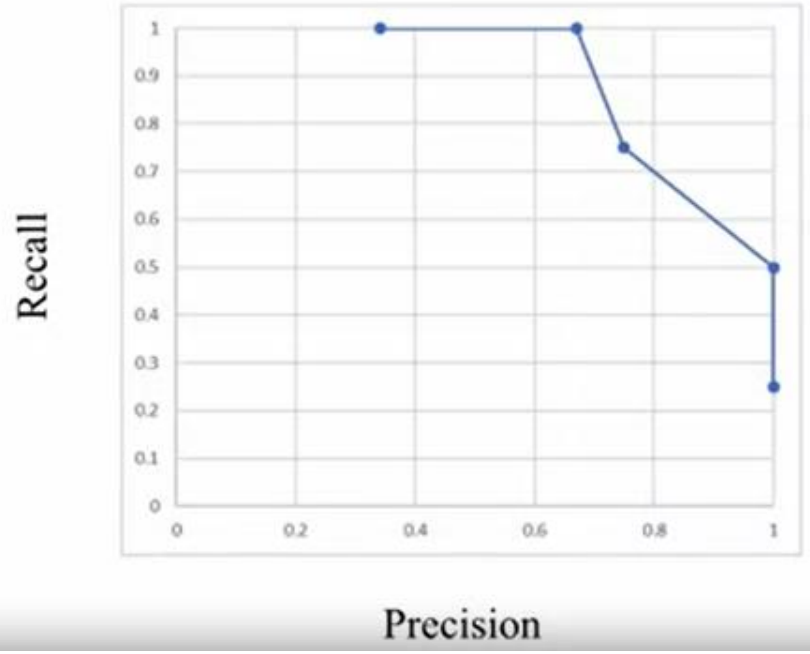
Predicted Probability of Class Membership (predict_proba)

- Typical rule: choose most likely class
  - e.g class 1 if threshold > 0.50.
- Adjusting threshold affects predictions of classifier.
- Higher threshold results in a more conservative classifier
  - e.g. only predict Class 1 if estimated probability of class 1 is above 70%
  - This increases precision. Doesn't predict class 1 as often, but when it does, it gets high proportion of class 1 instances correct.
- Not all models provide realistic probability estimates

Varying the Decision Threshold

True Label	Classifier score
0	-27.6457
0	-25.8486
0	-25.1011
0	-24.1511
0	-23.1765
0	-22.575
0	-21.8271
0	-21.7226
0	-19.7361
0	-19.5768
0	-19.3071
0	-18.9077
0	-13.5411
0	-12.8594
1	-3.9128
0	-1.9798
1	1.824
0	4.74931
1	15.234624
1	21.20597

Classifier score	Precision	Recall
-20	4/12=0.34	4/4=1.00
-10	4/6=0.67	4/4=1.00
0	3/4=0.75	3/4=0.75
10	2/2=1.0	2/4=0.50
20	1/1=1.0	1/4 = 0.25



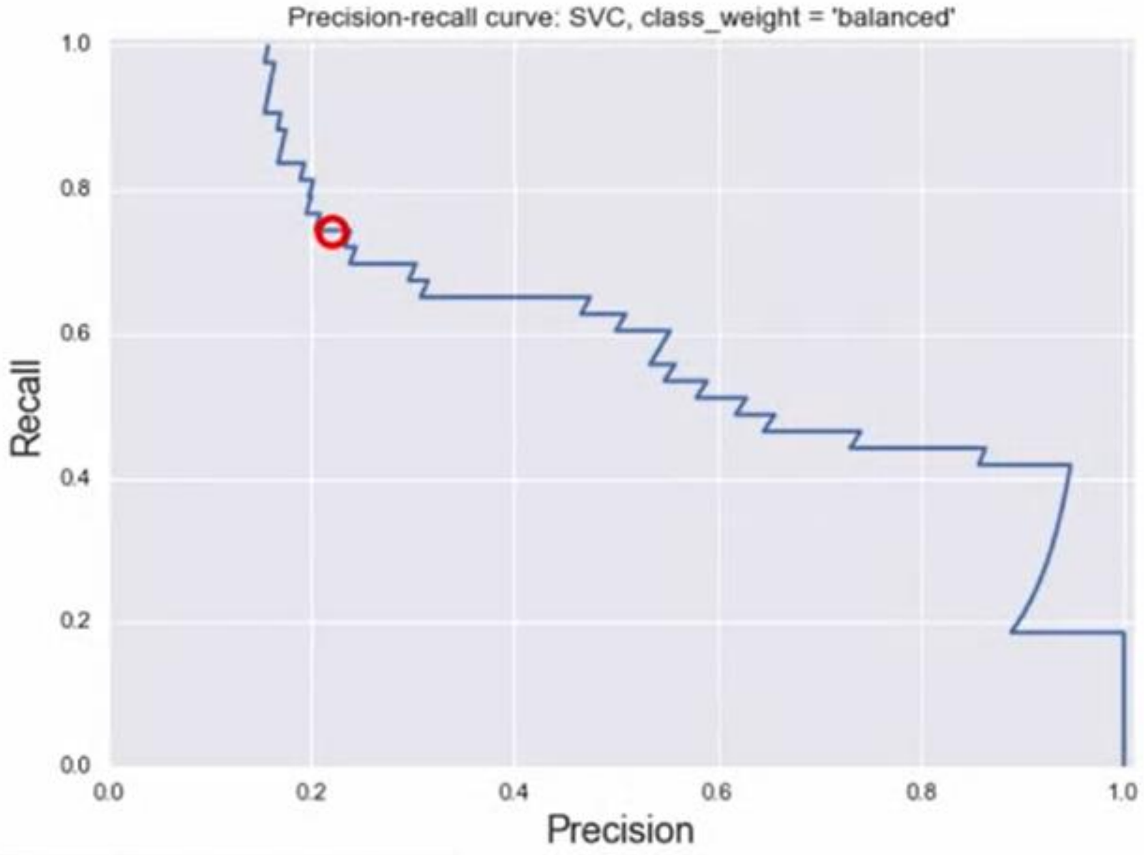
Precision-recall and ROC curves

Precision-Recall Curves

X-axis: Precision
Y-axis: Recall

Top right corner:
- The "ideal" point
- Precision = 1.0
- Recall = 1.0

"Steepness" of P-R curves is important:
- Maximize precision
- while maximizing recall

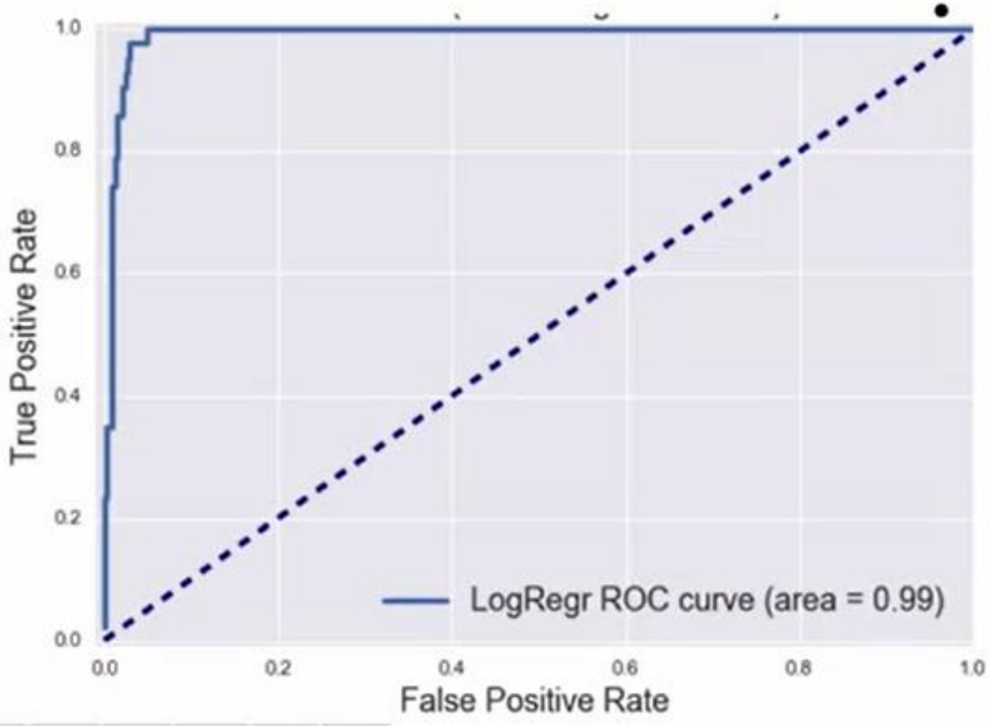


ROC Curves

X-axis: False Positive Rate
Y-axis: True Positive Rate

Top left corner:
- The "ideal" point
- False positive rate of zero
- True positive rate of one

"Steepness" of ROC curves is important:
- Maximize the true positive rate
- while minimizing the false positive rate



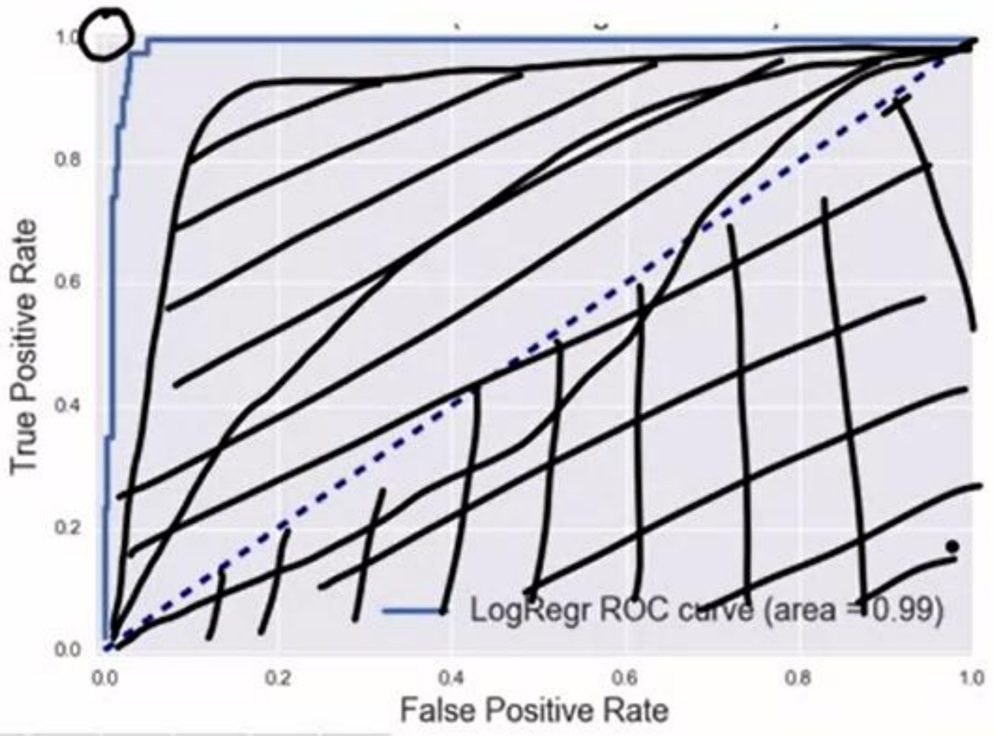
ROC Curves

AUC

X-axis: False Positive Rate
Y-axis: True Positive Rate

Top left corner:
- The "ideal" point
- False positive rate of zero
- True positive rate of one

"Steepness" of ROC curves is important:
- Maximize the true positive rate
- while minimizing the false positive rate



Multi-Class Evaluation

Multi-Class Evaluation

- **Multi-class evaluation is an extension of the binary case.**
  - A collection of true vs predicted binary outcomes, one per class
  - Confusion matrices are especially useful
  - Classification report
- **Overall evaluation metrics are averages across classes**
  - But there are different ways to average multi-class results: we will cover these shortly.
  - The support (number of instances) for each class is important to consider, e.g. in case of imbalanced classes
- **Multi-label classification: each instance can have multiple labels (not covered here)**

Multi-Class Confusion Matrix

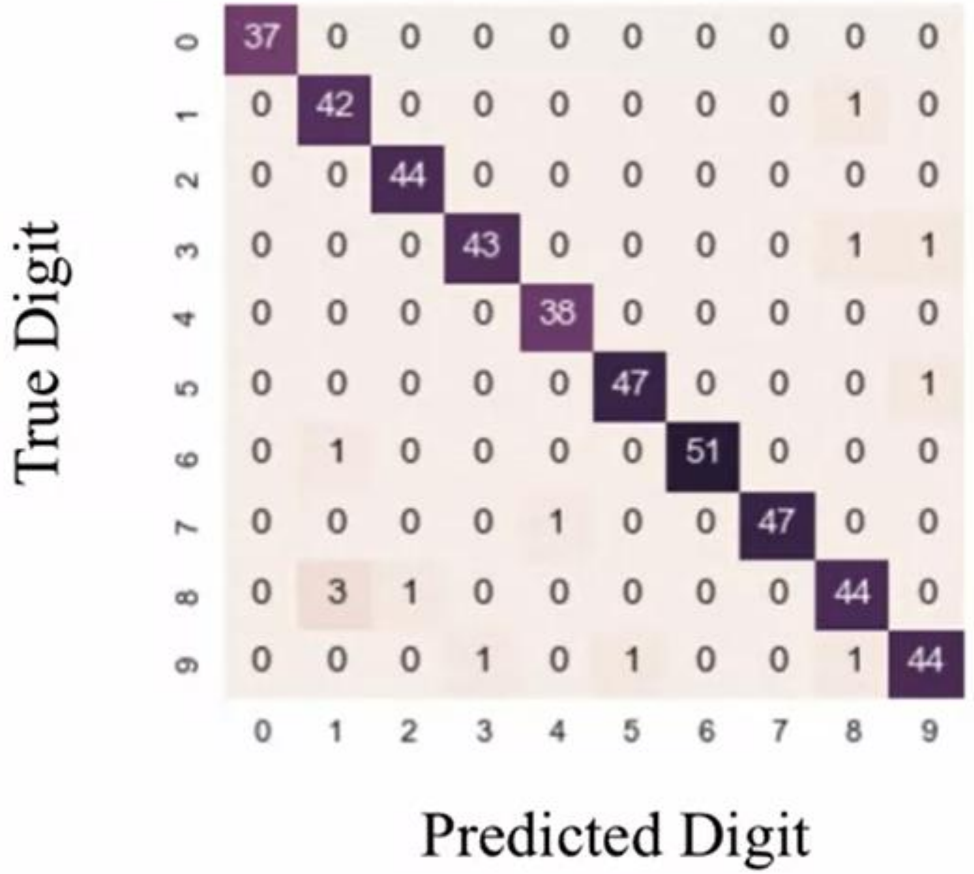


Figure 4

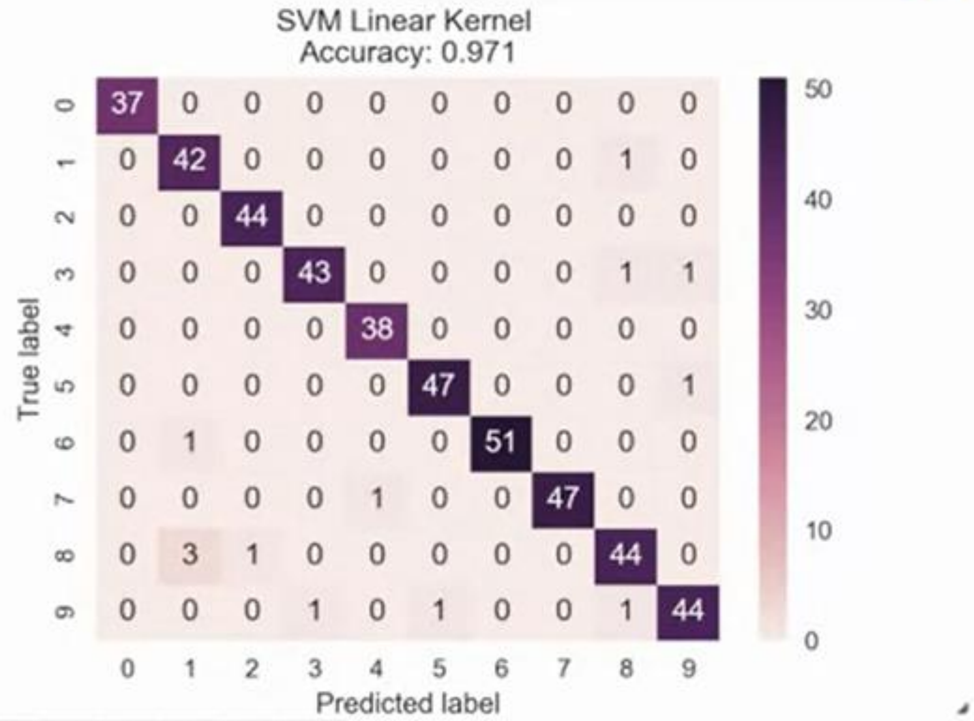
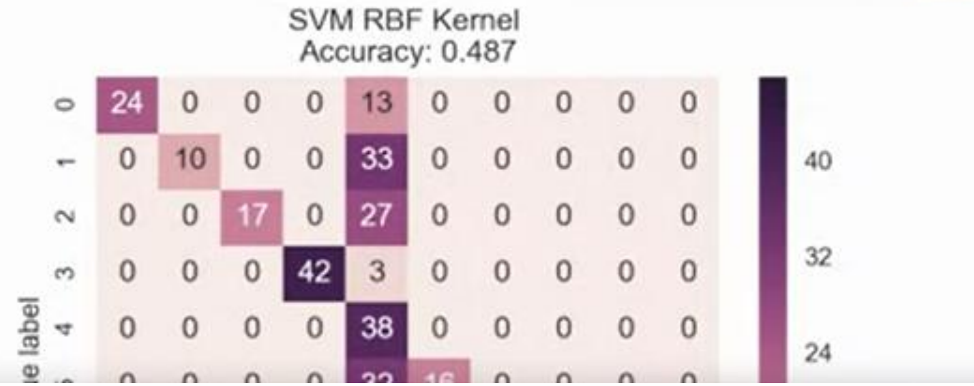


Figure 5



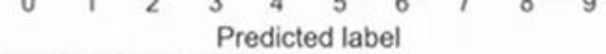
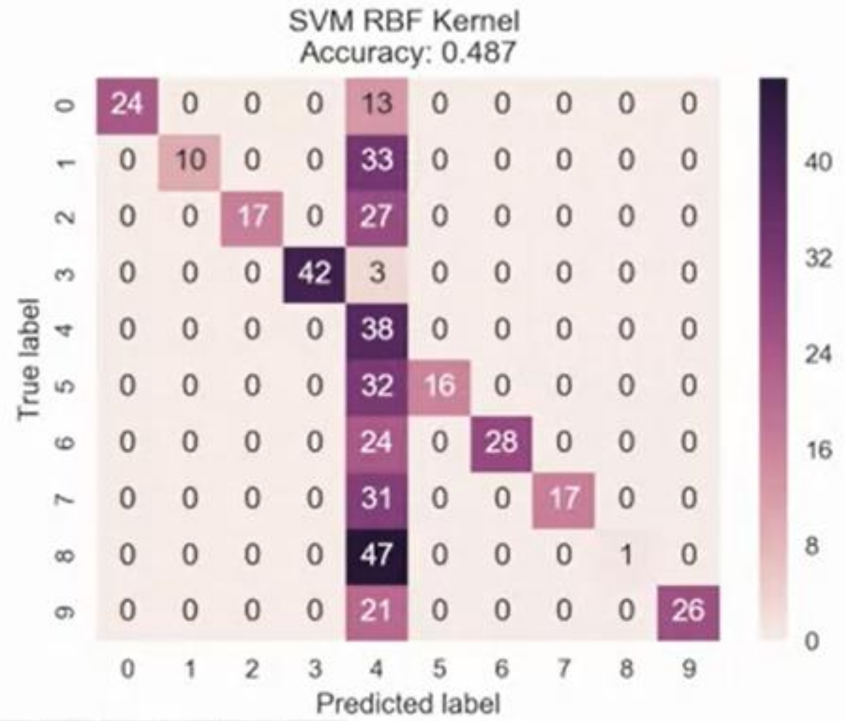


Figure 5




```
Out[21]: <matplotlib.text.Text at 0x119c6c780>
```

```
Out[21]: <matplotlib.text.Text at 0x119c6c780>
```

Multi-class classification report

```
In [22]: print(classification_report(y_test_mc, svm_predicted_mc))
```

```
             precision    recall  f1-score   support

          0       1.00      0.65      0.79        37
          1       1.00      0.23      0.38        43
          2       1.00      0.39      0.56        44
          3       1.00      0.93      0.97        45
          4       0.14      1.00      0.25        38
          5       1.00      0.33      0.50        48
          6       1.00      0.54      0.70        52
          7       1.00      0.35      0.52        48
          8       1.00      0.02      0.04        48
          9       1.00      0.55      0.71        47

avg / total       0.93      0.49      0.54       450
```

```
In [ ]: 
```

Micro vs Macro Average

Class	Predicted Class	Correct?
orange	lemon	0
orange	lemon	0
orange	apple	0
orange	orange	1
orange	apple	0
lemon	lemon	1
lemon	apple	0
apple	apple	1
apple	apple	1

Macro-average:

- Each class has equal weight.

1. Compute metric within each class
2. Average resulting metrics across classes

Class	Precision
orange	1/5 = 0.20
lemon	1/2 = 0.50
apple	2/2 = 1.00

Macro-average precision:
(0.20 + 0.50 + 1.00) / 3 = **0.57**

Micro vs Macro Average

Class	Predicted Class	Correct?
orange	lemon	0
orange	lemon	0
orange	apple	0
orange	orange	1
orange	apple	0
lemon	lemon	1
lemon	apple	0
apple	apple	1
apple	apple	1

Micro-average:

- Each instance has equal weight.
- Largest classes have most influence

1. Aggregrate outcomes across all classes
2. Compute metric with aggregate outcomes

Micro-average precision:
4 / 9 = **0.44**

Macro-Average vs Micro-Average

- If the classes have about the same number of instances, macro- and micro-average will be about the same.
- If some classes are much larger (more instances) than others, and you want to:
  - Weight your metric toward the largest ones, use micro-averaging.
  - Weight your metric toward the smallest ones, use macro-averaging.
- If the micro-average is much lower than the macro-average then examine the larger classes for poor metric performance.
- If the macro-average is much lower than the micro-average then examine the smaller classes for poor metric performance.

Regression Evaluation

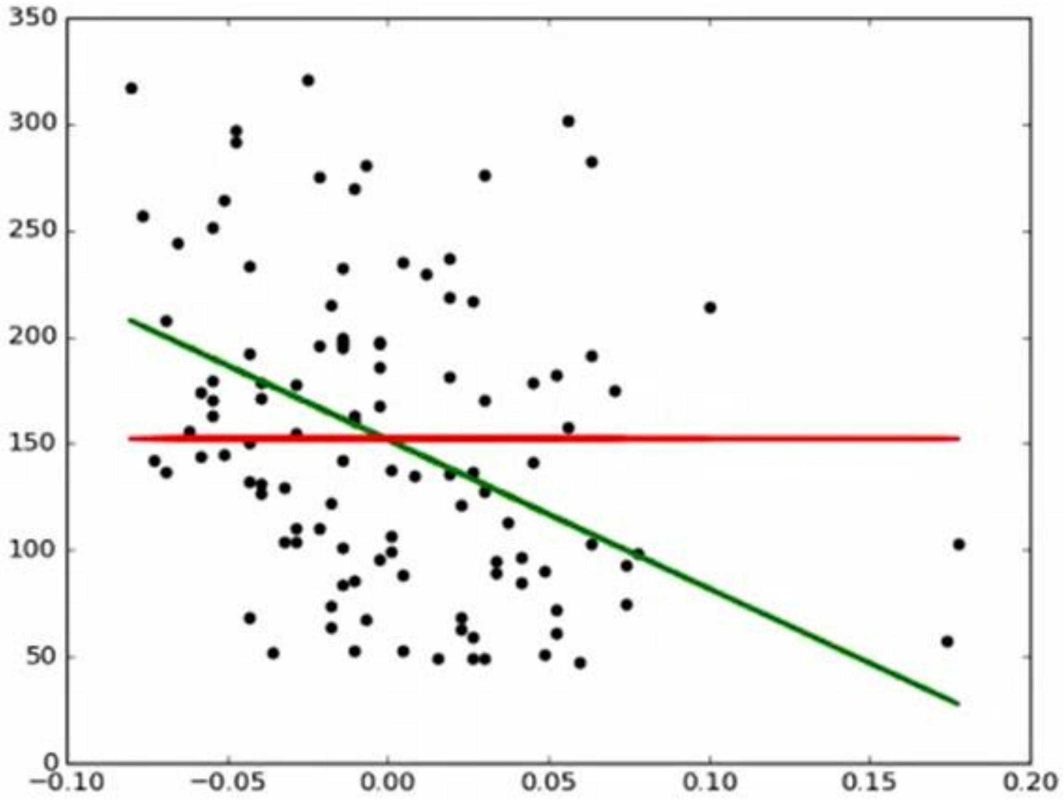
Regression Metrics

- **Typically r2_score is enough**
  - *Reminder: computes how well future instances will be predicted*
  - *Best possible score is 1.0*
  - *Constant prediction score is 0.0*
- **Alternative metrics include:**
  - *mean_absolute_error (absolute difference of target & predicted values)*
  - *mean_squared_error (squared difference of target & predicted values)*
  - *median_absolute_error (robust to outliers)*

Dummy Regressors

As in classification, comparison to a 'dummy' prediction model that uses a fixed rule can be useful.

For this, scikit.learn provides dummy regressors.

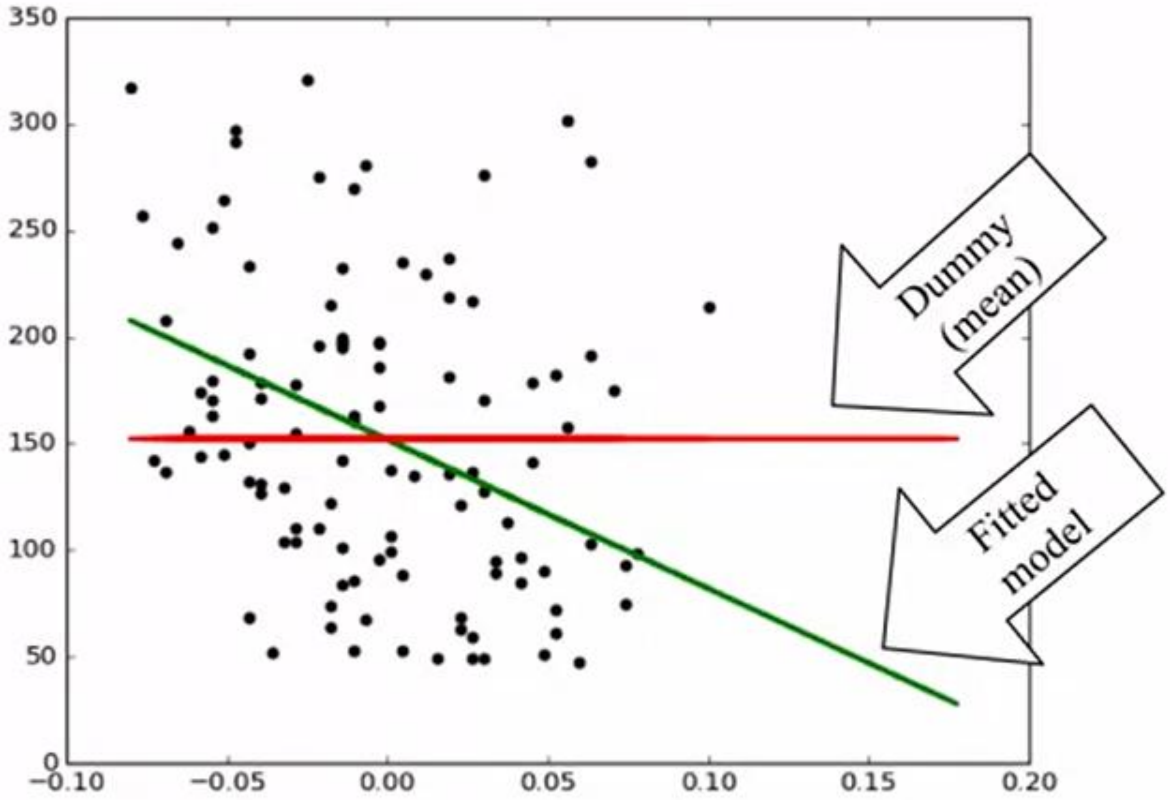


Dummy Regressors

As in classification, comparison to a 'dummy' prediction model that uses a fixed rule can be useful.

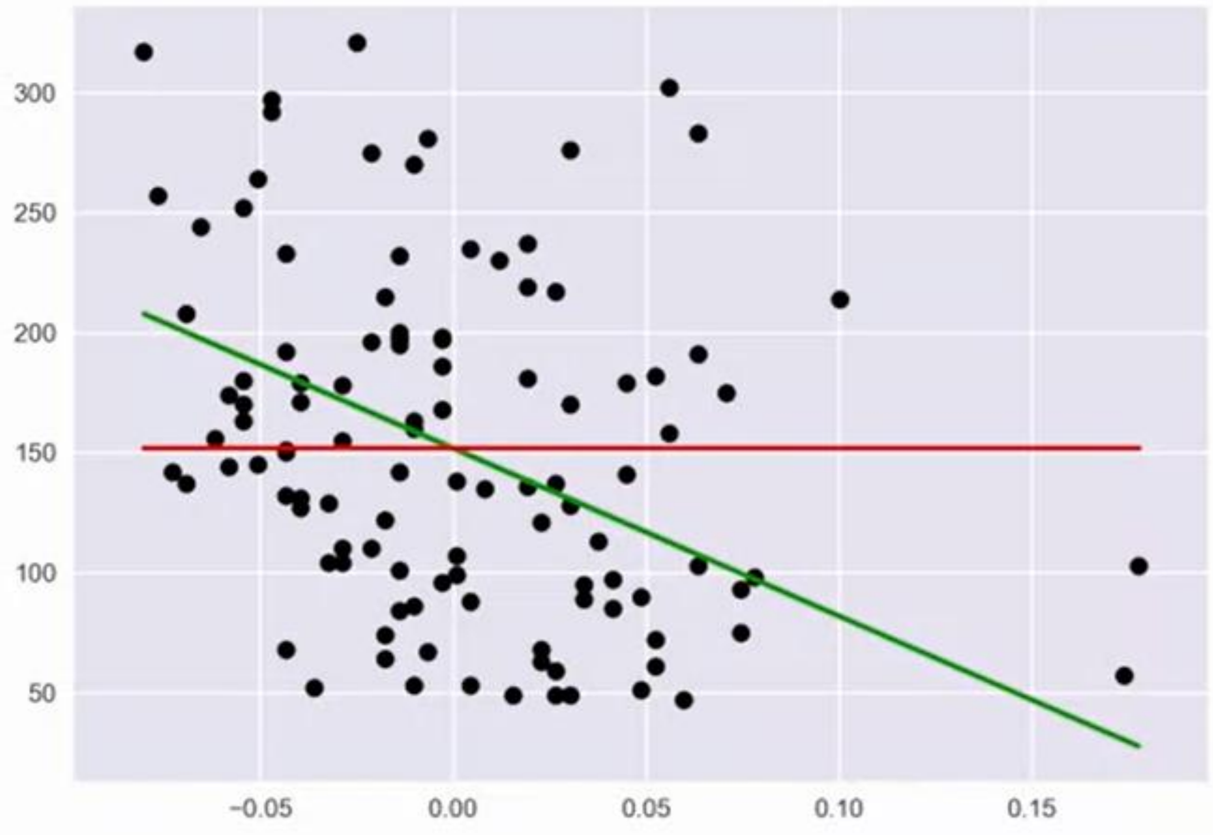
For this, scikit.learn provides dummy regressors

```
Linear model, coefficients: [-698.80206267]
Mean squared error (dummy): 4965.13
Mean squared error (linear model): 4646.74
r2_score (dummy): -0.00
r2_score (linear model): 0.06
```



```
Linear model, coefficients:  [-698.80206267]
Mean squared error (dummy): 4965.13
Mean squared error (linear model): 4646.74
r2_score (dummy): -0.00
r2_score (linear model): 0.06
```

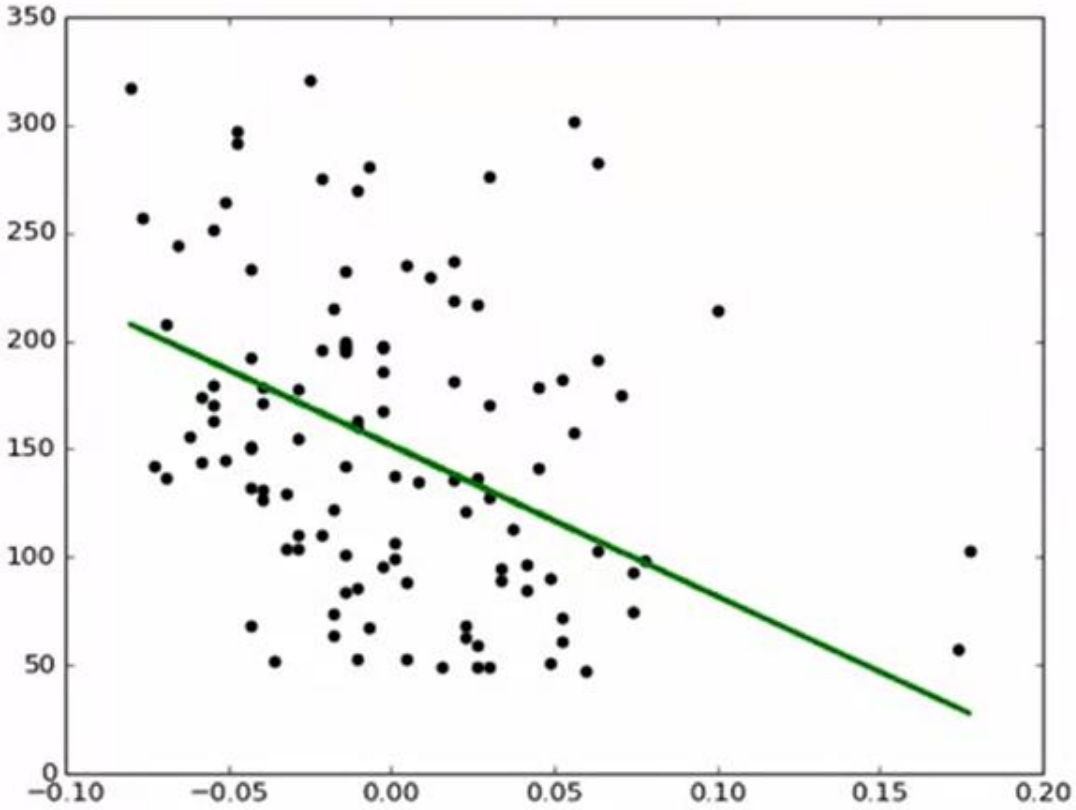
Figure 1



Dummy Regressors

The DummyRegressor class implements four simple baseline rules for regression, using the `strategy` parameter:

- `mean` predicts the mean of the training target values.
- `median` predicts the median of the training target values.
- `quantile` predicts a user-provided quantile of the training target values (e.g. value at the 75th percentile)
- `constant` predicts a custom constant value provided by the user.

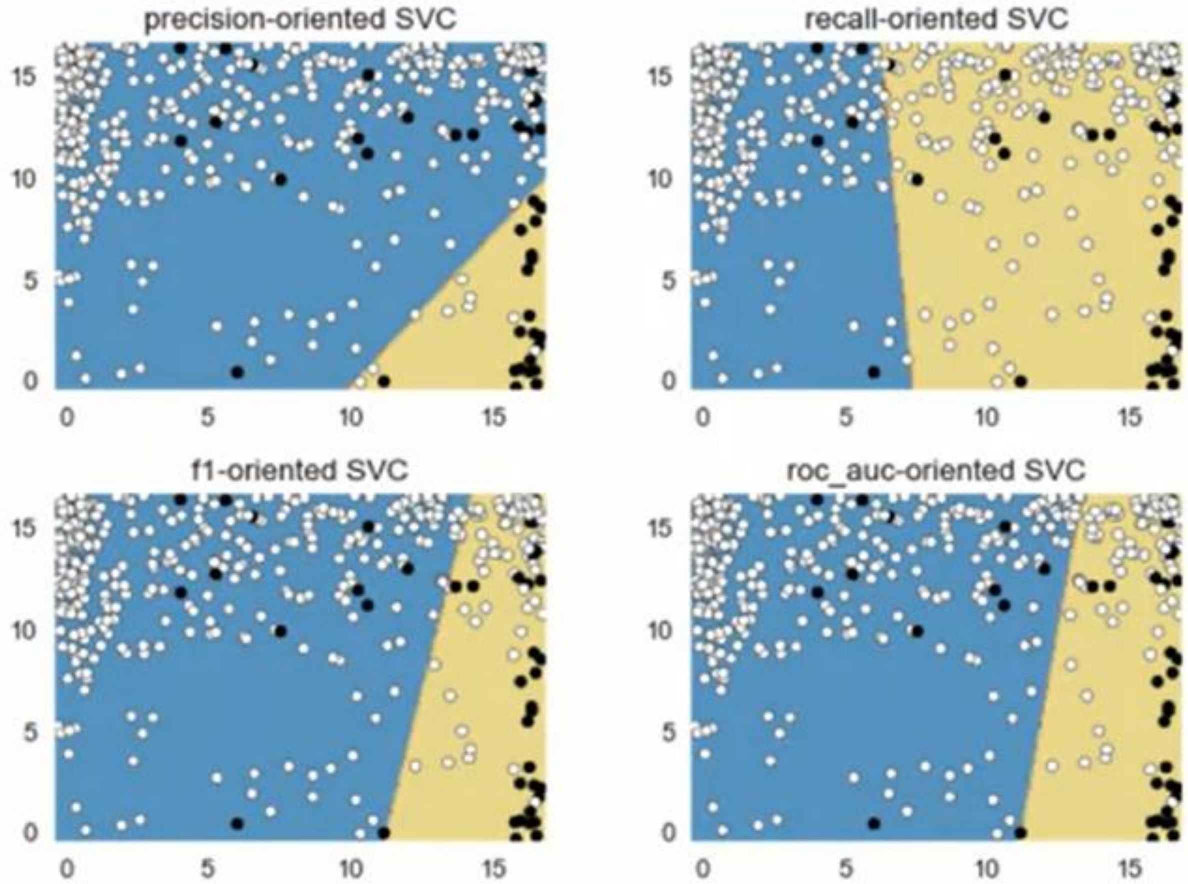


Optimizing Classifiers for Different Metrics

Model Selection Using Evaluation Metrics

- **Train/test on same data**
  - Single metric.
  - Typically overfits and likely won't generalize well to new data.
  - But can serve as a sanity check: low accuracy on the training set may indicate an implementation problem.
- **Single train/test split**
  - Single metric.
  - Speed and simplicity.
  - Lack of variance information
- **K-fold cross-validation**
  - K train-test splits.
  - Average metric over all splits.
  - Can be combined with parameter grid search: GridSearchCV (def. cv = 3)

Example: Optimizing a Classifier Using Different Evaluation Metrics



Training, Validation, and Test Framework for Model Selection and Evaluation

- Using only cross-validation or a test set to do model selection may lead to more subtle overfitting / optimistic generalization estimates
- Instead, use three data splits:
  1. Training set (model building)
  2. Validation set (model selection)
  3. Test set (final evaluation)
- In practice:
  - Create an initial training/test split
  - Do cross-validation on the training data for model/parameter selection
  - Save the held-out test set for final model evaluation

Concluding Notes

- **Accuracy is often not the right evaluation metric for many real-world machine learning tasks**
  - False positives and false negatives may need to be treated very differently
  - Make sure you understand the needs of your application and choose an evaluation metric that matches your application, user, or business goals.
- **Examples of additional evaluation methods include:**
  - Learning curve: How much does accuracy (or other metric) change as a function of the amount of training data?
  - Sensitivity analysis: How much does accuracy (or other metric) change as a function of key learning parameter values?