FRAUD DETECTION IN PYTHON

Review of classification methods for fraud detection

Charlotte Werger
Data Scientist

What is classification?

Goal of classification: Use known fraud cases to train a model to recognise new fraud cases

Examples:

- Email Spam/Not spam
- Transaction online fraudulent Yes/No
- Tumor Malignant/Benign?

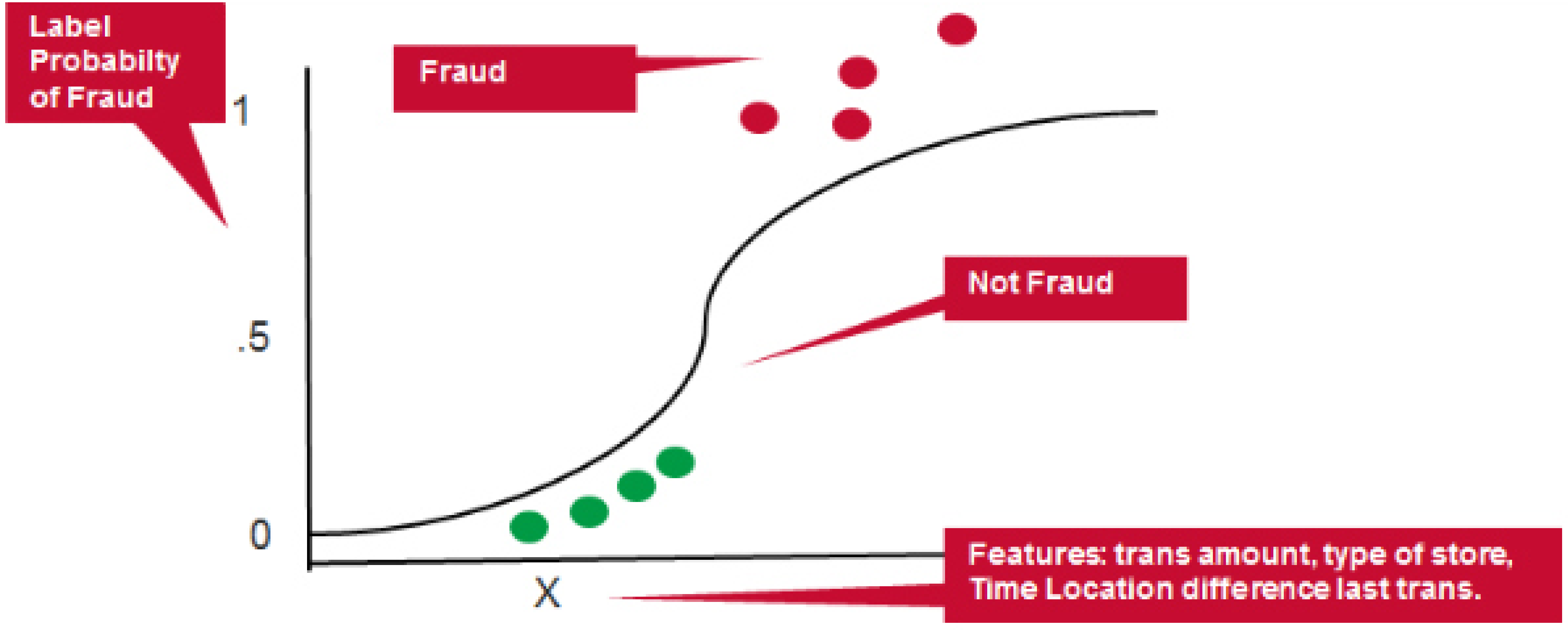
Variable to predict: $y \in 0, 1$

0: Negative class ("majority" normal cases)

1: Positive class ("minority" fraud cases)

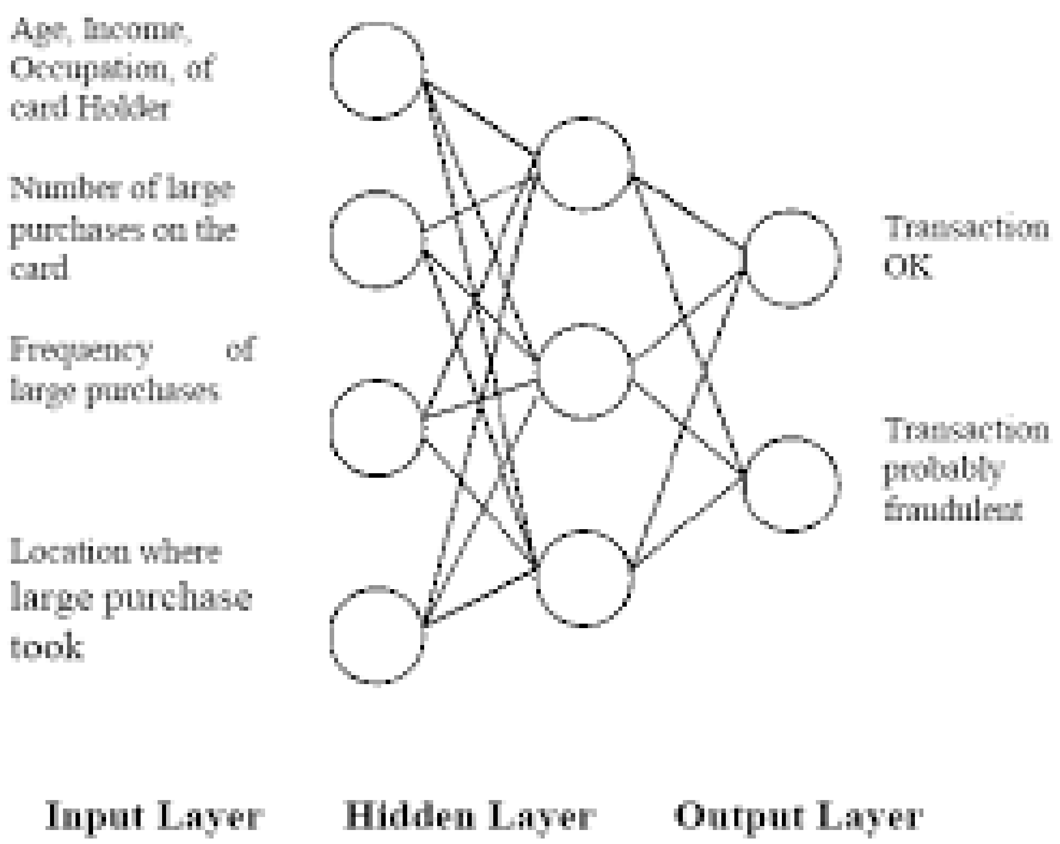
Classification methods commonly used for fraud detection

- Logistic Regression



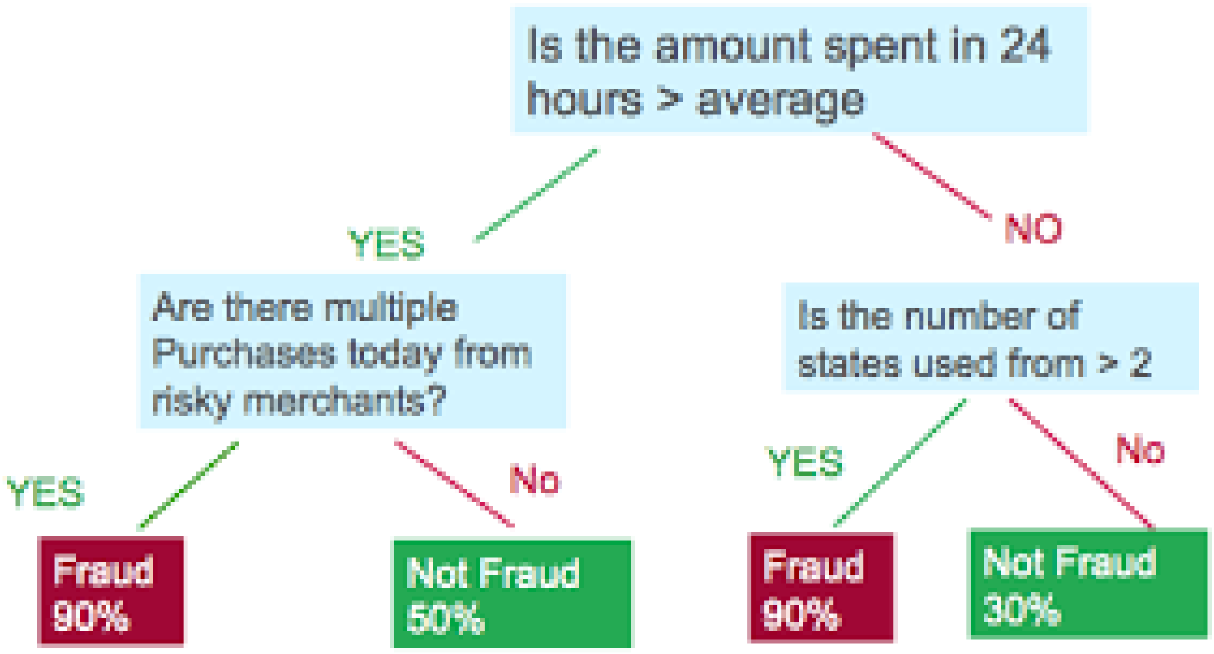
Classification methods commonly used for fraud detection

- Neural Network



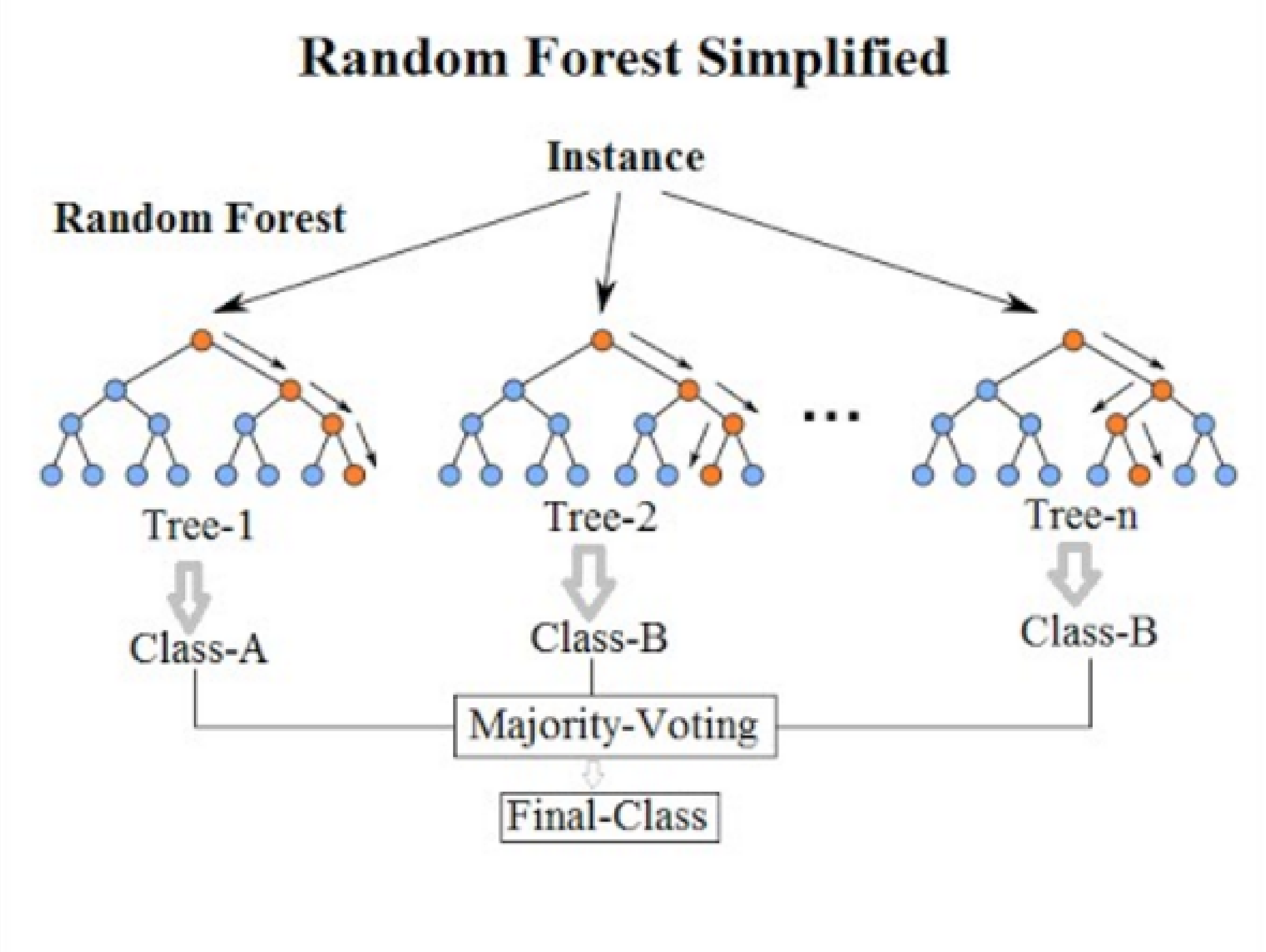
Classification methods commonly used for fraud detection

- Decision trees
- Random Forests



Decision Trees and Random Forests

- Random forests are a collection of trees on random subsets of features



Random Forests for fraud detection

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier(random_state=42)
```

```
model.fit(X_train, y_train)
```

```
predicted = model.predict(X_test)
```

```
print (metrics.accuracy_score(y_test, predicted))

0.991324200913242
```

FRAUD DETECTION IN PYTHON

Let's practice!

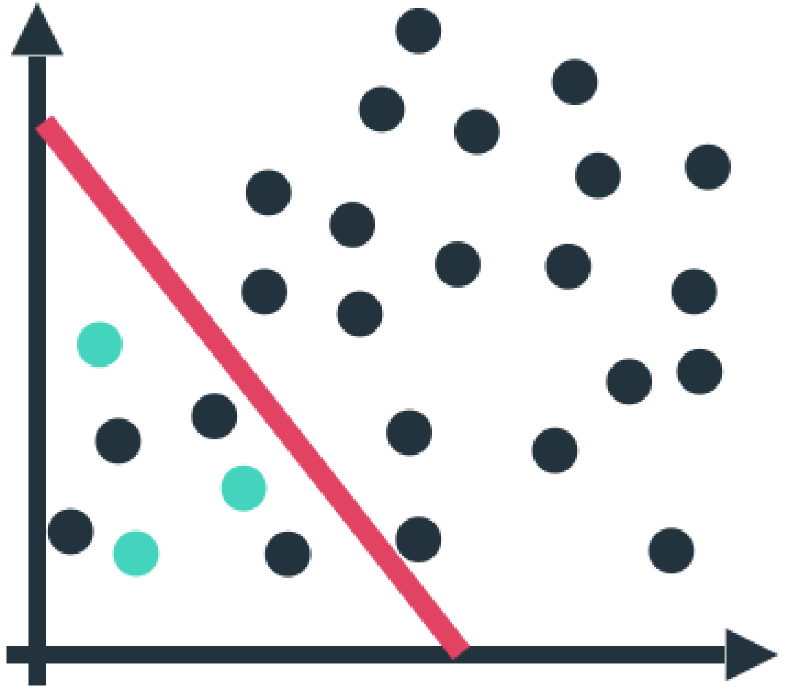
FRAUD DETECTION IN PYTHON

Measuring fraud detection performance

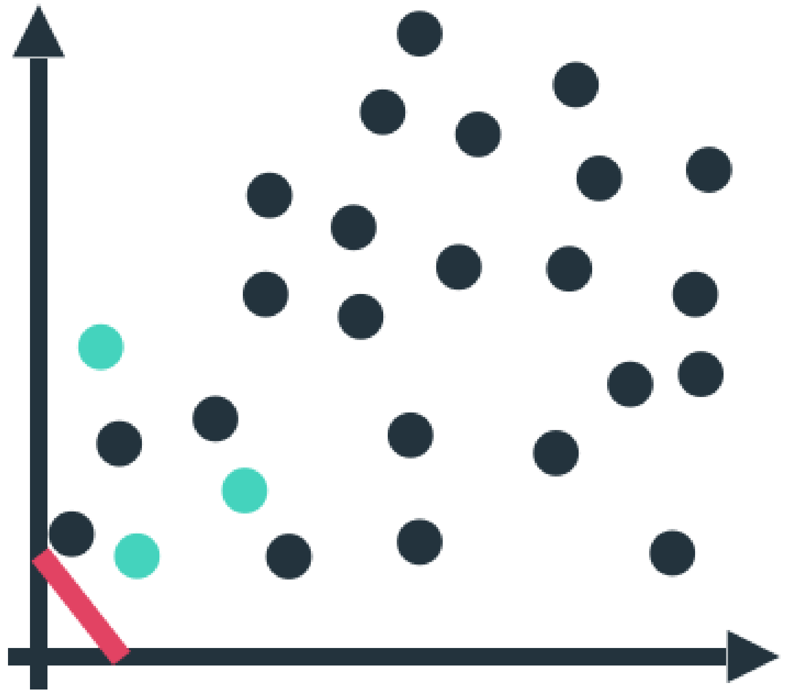
Charlotte Werger
Data Scientist

Accuracy isn't everything

Throw accuracy out of the window when working on fraud detection problems

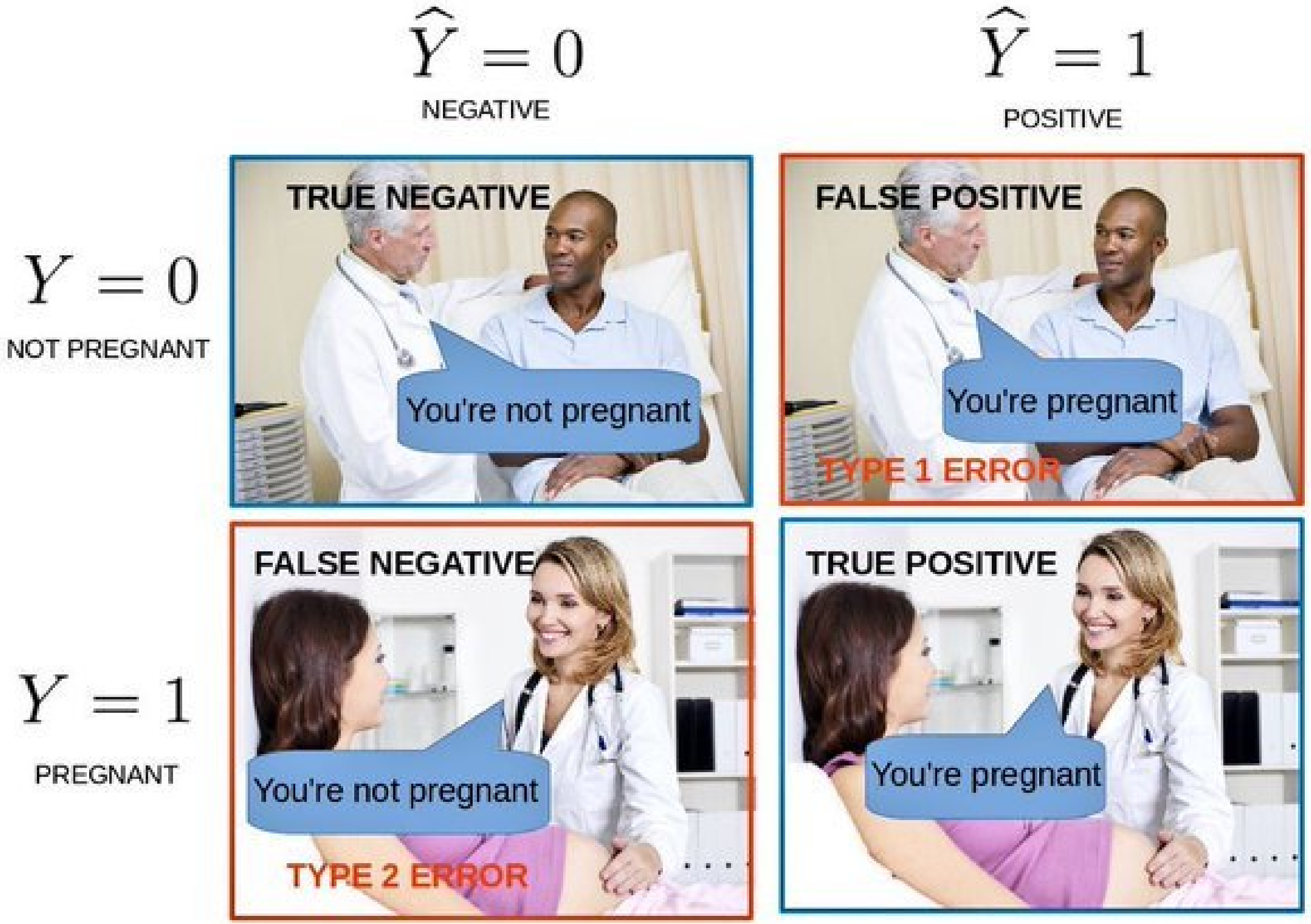


Accuracy = 93%

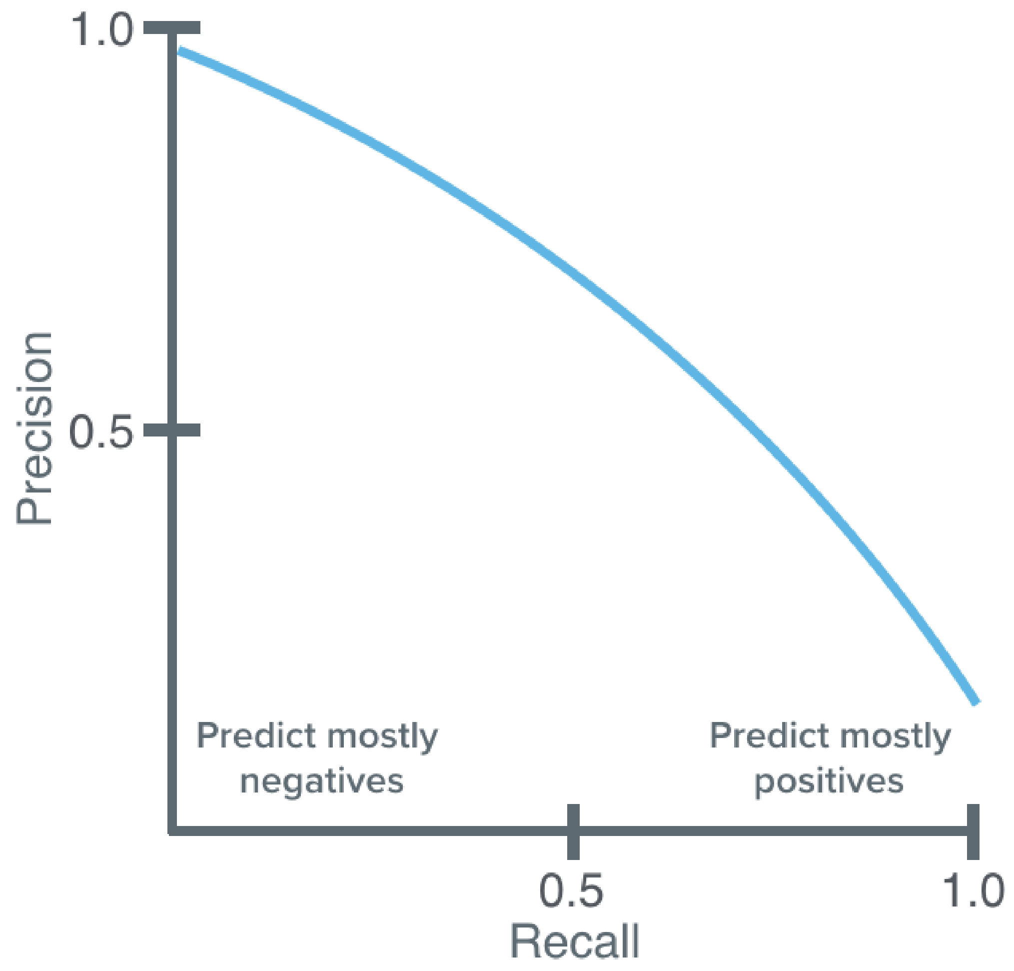


Accuracy = 97%

False positives, false negatives and actual fraud caught



Precision Recall trade-off



$$Precision = \frac{\#True\ Positives}{\#True\ Positives + \#False\ Positives}$$

$$Recall = \frac{\#True\ Positives}{\#True\ Positives + \#False\ Negatives}$$

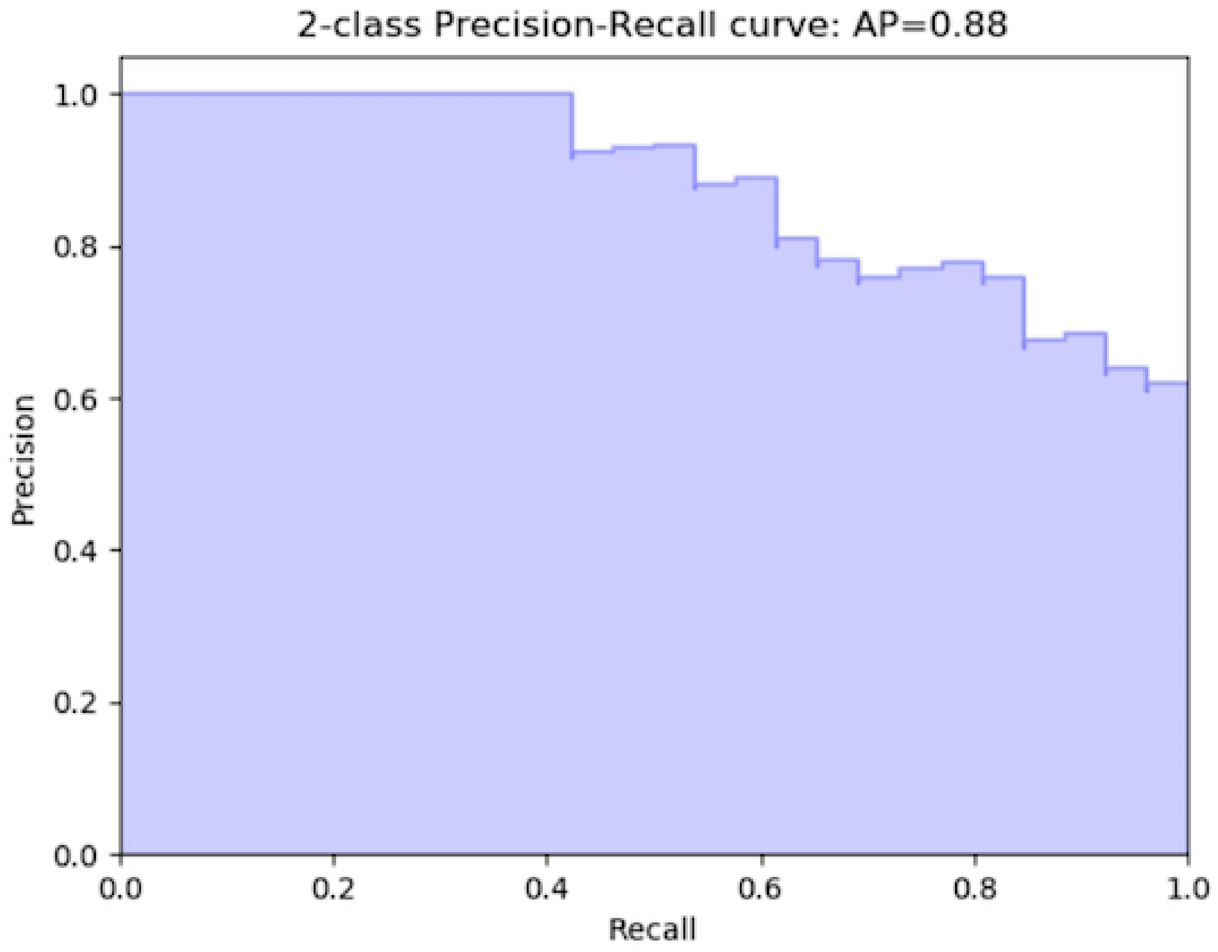
$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$= \frac{2 \times TP}{2 \times TP + FP + FN}$$

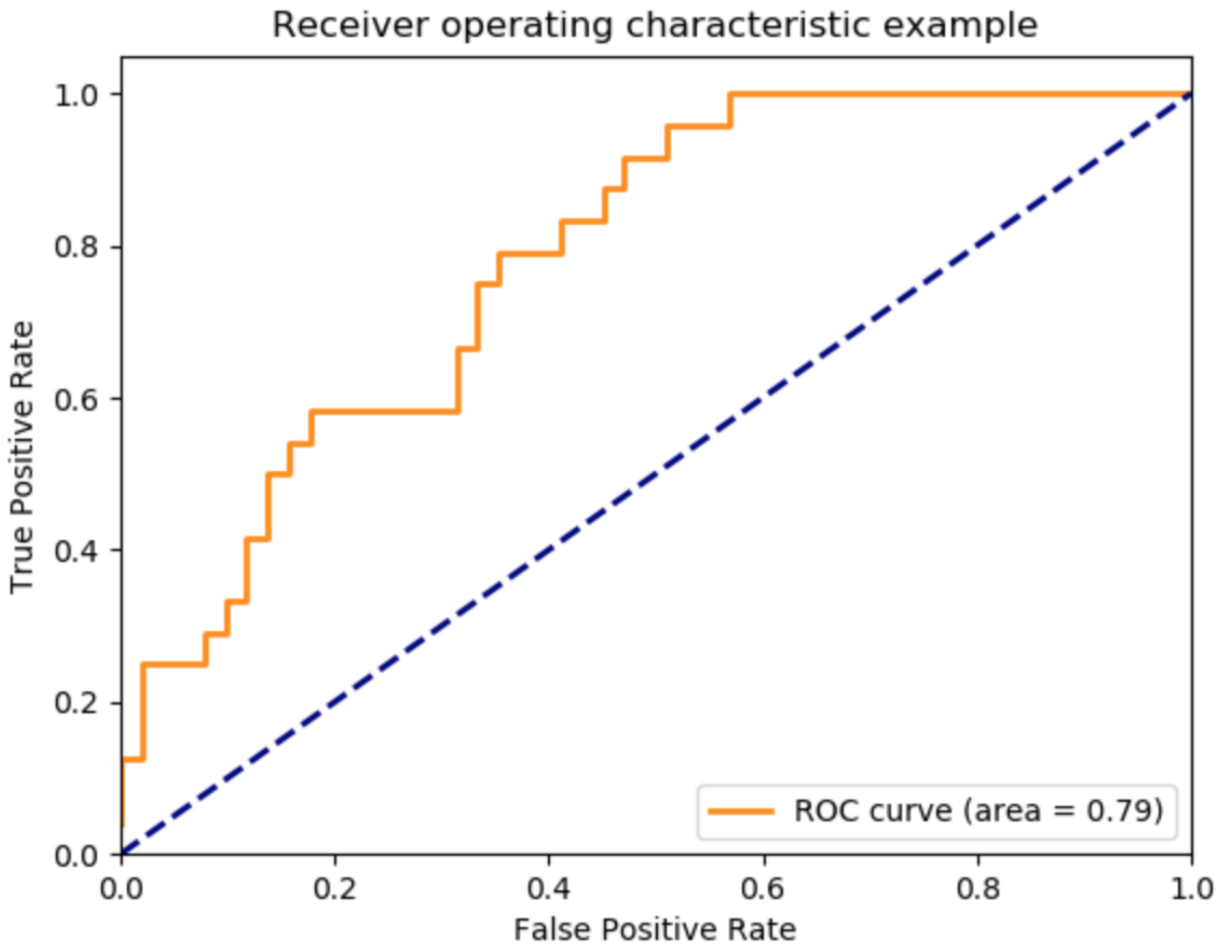
Obtaining performance metrics

```
# Import the packages
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score

# Calculate average precision and the PR curve
average_precision = average_precision_score(y_test, predicted)

# Obtain precision and recall
precision, recall, _ = precision_recall_curve(y_test, predicted)
```

Precision-Recall Curve



ROC curve to compare algorithms




```
# Obtain model probabilities
probs = model.predict_proba(X_test)

# Print ROC_AUC score using probabilities
print(metrics.roc_auc_score(y_test, probs[:, 1]))
```

Confusion matrix and classification report

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
# Obtain predictions
predicted = model.predict(X_test)
```

```
# Print classification report using predictions
print(classification_report(y_test, predicted))

  precision    recall  f1-score   support

        0.0       0.99      1.00      1.00      2099
        1.0       0.96      0.80      0.87        91

avg / total       0.99      0.99      0.99      2190
```

```
# Print confusion matrix using predictions
print(confusion_matrix(y_test, predicted))

[[2096    3]
 [  18   73]]
```

FRAUD DETECTION IN PYTHON

Let's practice!

FRAUD DETECTION IN PYTHON

Adjusting your algorithms for fraud detection

Charlotte Werger
Data Scientist

Balanced weights

```
model = RandomForestClassifier(class_weight='balanced')
```

```
model = RandomForestClassifier(class_weight='balanced_subsample')
```

```
model = LogisticRegression(class_weight='balanced')
```

```
model = SVC(kernel='linear', class_weight='balanced', probability=True)
```

Hyperparameter tuning for fraud detection

```
model = RandomForestClassifier(class_weight={0:1,1:4},random_state=1)

model = LogisticRegression(class_weight={0:1,1:4}, random_state=1)
```

```
model = RandomForestClassifier(n_estimators=10,
criterion='gini',
max_depth=None,
min_samples_split=2,
min_samples_leaf=1,
max_features='auto',
n_jobs=-1, class_weight=None)
```

Using GridSearchCV

```
from sklearn.model_selection import GridSearchCV
```

```
# Create the parameter grid
param_grid = {
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
```

```
# Define which model to use
model = RandomForestRegressor()
```

```
# Instantiate the grid search model
grid_search_model = GridSearchCV(estimator = model,
param_grid = param_grid, cv = 5,
n_jobs = -1, scoring='f1')
```

Finding the best model with GridSearchCV

```
# Fit the grid search to the data
grid_search_model.fit(X_train, y_train)
```

```
# Get the optimal parameters
grid_search_model.best_params_

{'bootstrap': True,
 'max_depth': 80,
 'max_features': 3,
 'min_samples_leaf': 5,
 'min_samples_split': 12,
 'n_estimators': 100}
```

```
# Get the best_estimator results
grid_search.best_estimator_
grid_search.best_score_
```

FRAUD DETECTION IN PYTHON

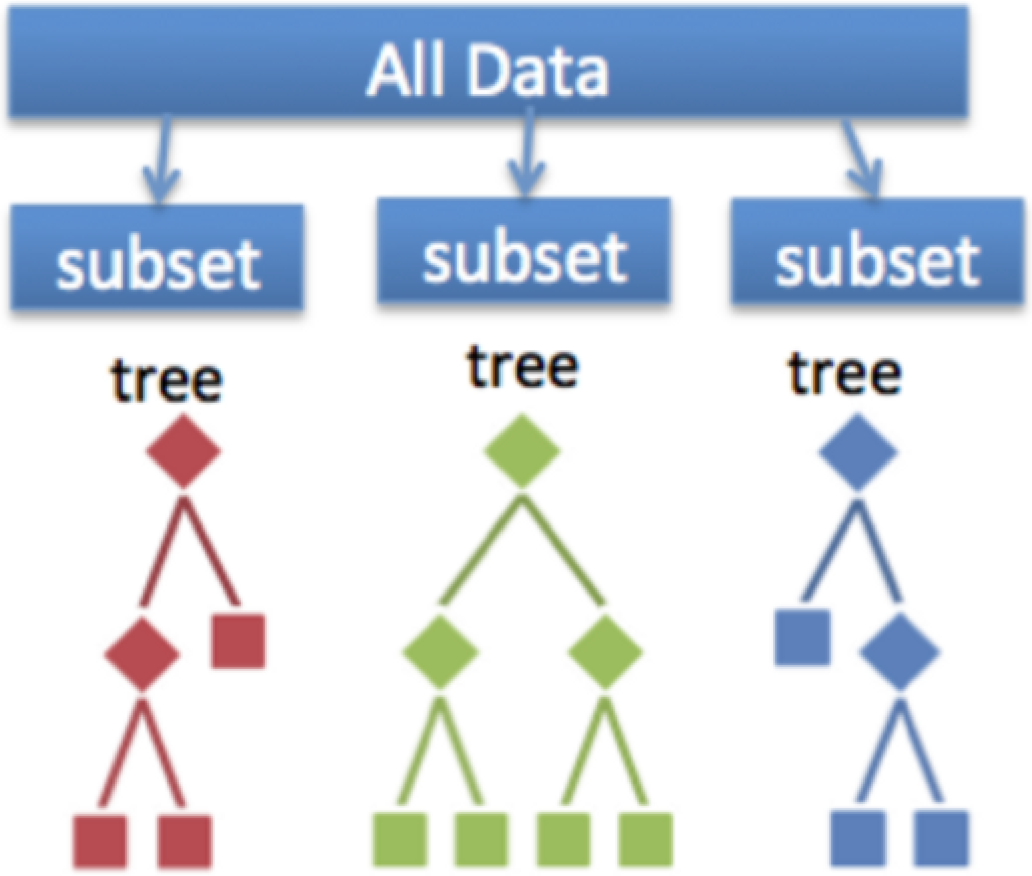
Let's practice!

FRAUD DETECTION IN PYTHON

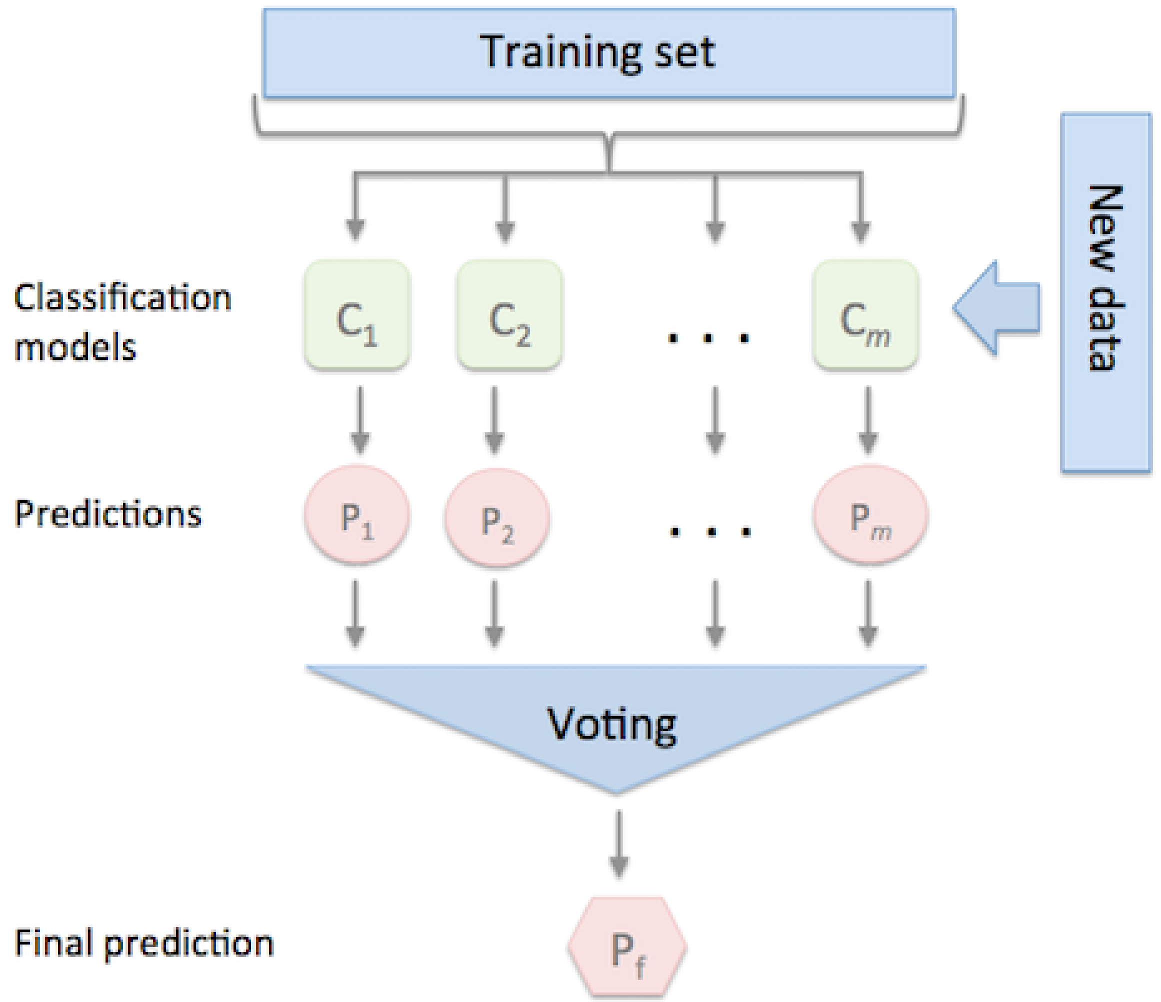
Using ensemble methods to improve fraud detection

Charlotte Werger
Data Scientist

What are Ensemble Methods: Bagging versus Stacking



Stacking Ensemble Methods



Why use ensemble methods for fraud detection

Ensemble methods:

- Are robust
- Can help you avoid overfitting
- Can typically improve prediction performance
- Are a winning formula at prestigious Kaggle competitions

Voting Classifier

```
from sklearn.ensemble import VotingClassifier

clf1 = LogisticRegression(random_state=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
```

```
ensemble_model = VotingClassifier(estimators=[('lr', clf1),
('rf', clf2), ('gnb', clf3)], voting='hard')
```

```
ensemble_model.fit(X_train, y_train)
ensemble_model.predict(X_test)
```

```
VotingClassifier(estimators=[('lr', clf1), ('rf', clf2),
('gnb', clf3)], voting='soft', weights=[2,1,1])
```

Reliable labels for fraud detection

FRAUD DETECTION IN PYTHON

Let's practice