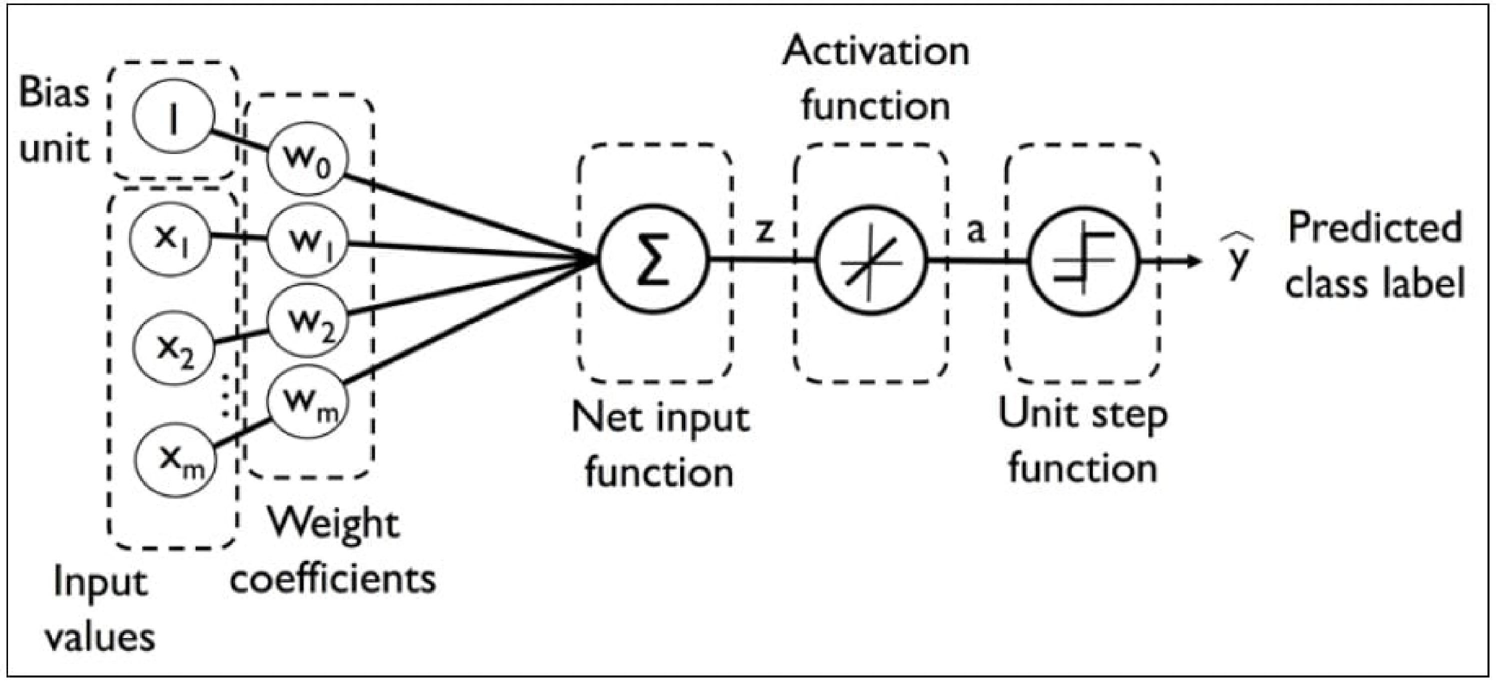
Introduction to deep learning

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

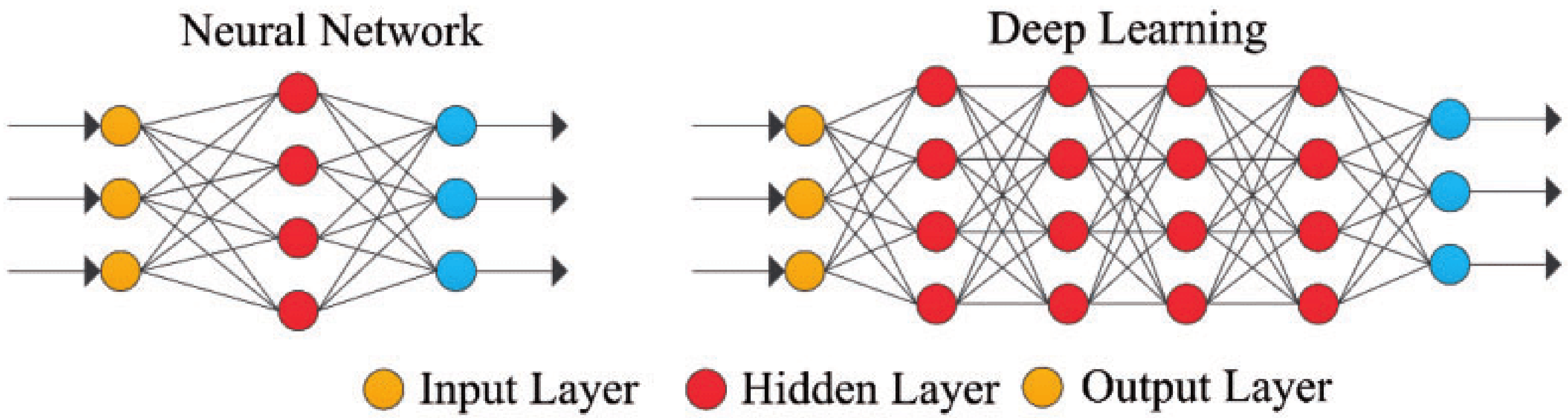
Kevin Huo
Instructor

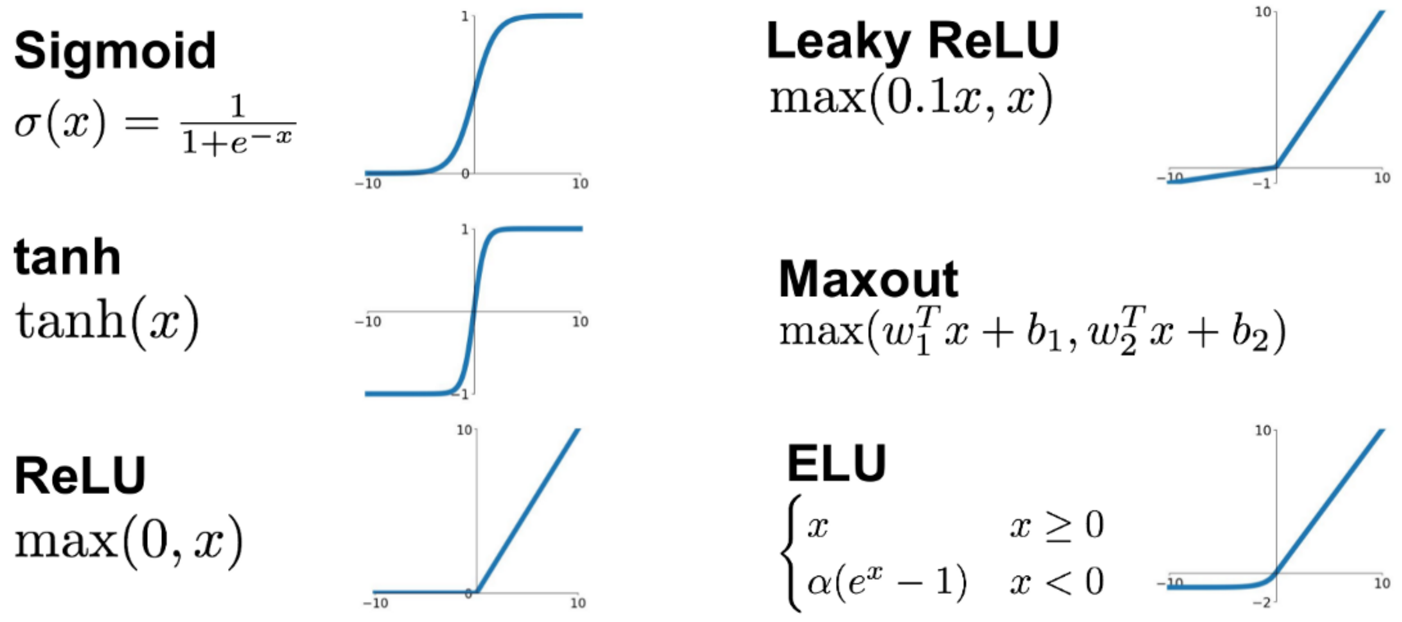
Perceptrons



- Input features are standardized
- Inputs get summed through weights
- Output goes through activation function
- Unit step function to convert output into predicted class

Hidden layers and activation functions





Implementation

```
clf = MLPClassifier()
print(clf)
```

```
MLPClassifier(activation='relu',
              alpha=0.0001,
              ...
              hidden_layer_sizes=(100,),
              learning_rate = 'constant',
              ...
              max_iter=200,
              ...)
```

Other considerations

- Standardization is important before usage
  - `X = StandardScaler().fit_transform(X)`
- Very large networks with many millions of parameters
  - Feature matrices are often "sparse"
- Better performance with more data
  - However, downside is less transparency and longer compute time

Let's practice!

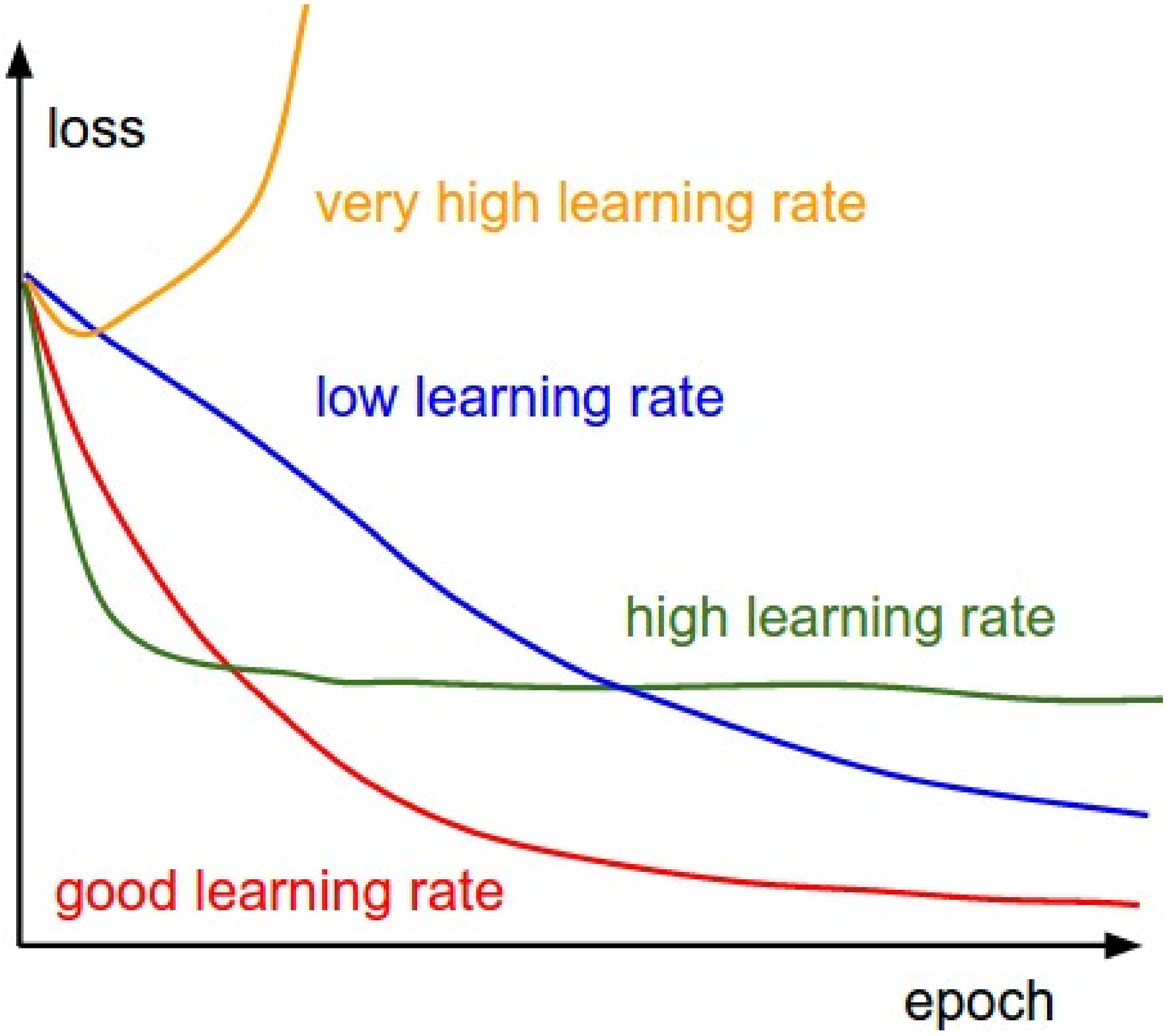
PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Hyperparameter tuning in deep learning

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

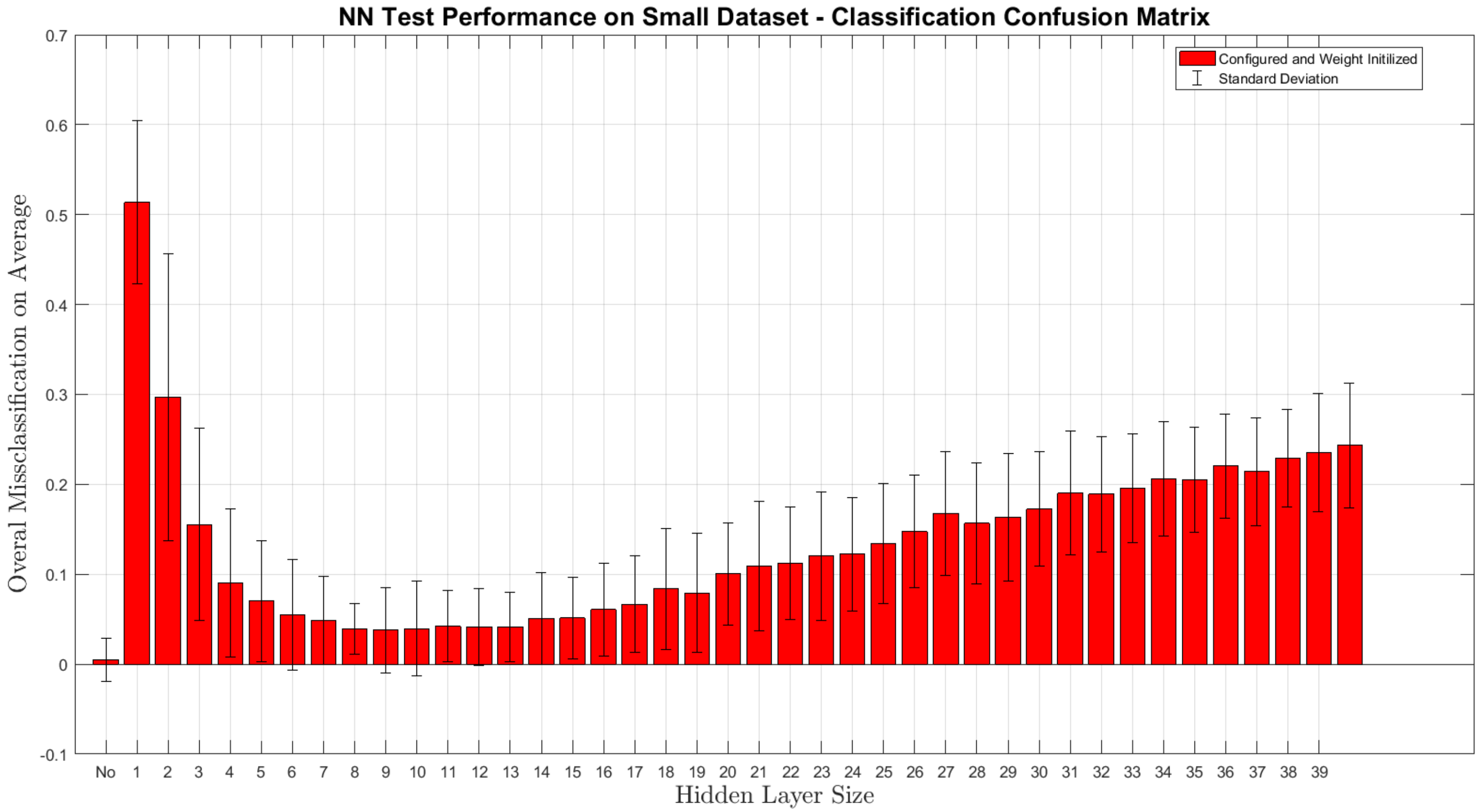
Kevin Huo
Instructor

Learning rate and number of iterations



- Weights are updated iteratively
  - Uses back-propagation
- A good learning rate will result in loss dropping quickly and stabilizing
  - Shown in red line
- Too high of a learning rate will result in an "overshoot" and very high loss
  - Shown in yellow line

Choosing hidden layers



- Increase in performance up to certain level of complexity, then drop-off afterwards.

Grid search

```
param_grid = {'max_iter': [10, 20],
              'hidden_layer_sizes': [(8, ), (16, )]}
```

```
clf = GridSearchCV(
  estimator = MLPClassifier(), param_grid = param_grid,
  n_jobs = 4)
print(clf.best_score_)
print(clf.best_estimator_)
```

```
0.65
MLPClassifier(hidden_layer_size = (16,), ...)
```

Real life extensions

- Batch size and epochs are also potential hyperparameters
  - Batch size is for mini-batch (training is done in small batches), and epochs are for the number of iterations through whole training data
- Initialization of weights can vary and affect results
  - Examples of different initializations: uniformly distributed, normally distributed, etc.
- Keras and Tensorflow are often used rather than `sklearn`
  - This is due to limited functionality on `sklearn` in comparison

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Model evaluation

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

Precision and recall

- Precision: ROI on ad spend through clicks
  - Low precision means very little tangible ROI on clicks
- Recall: targeting relevant audience
  - Low recall means missed out opportunities on ROI
- It may be sensible to weight the two differently
  - Companies are likely to care more about avoiding low precision compared to low recall

F-beta score

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

- Beta coefficient: represents relative weighting of two metrics
    - Beta between 0 and 1 means precision is made smaller and hence weighted more, whereas beta > 1 means precision is made larger and hence weighted less
- Implementation available in `sklearn` via: `fbeta_score(y_true, y_pred, beta)`
    - `y_true` is true targets and `y_pred` the predicted targets

AUC of ROC curve versus precision

```
roc_auc = roc_auc_score(y_test, y_score[:, 1])
```

```
fpr = 1 - tn / (tn + fp)
precision = tp / (tp + fp)
```

- Imbalanced dataset: `fpr` can be low when `precision` is also low.
- Let us assume we have 100 TN, and 10 TP and 10 FP.

```
fpr = 1 - 100 / (100 + 10) = 0.091
precision = tp / (tp + fp) = 0.5
```

- Low FPR can lead to high AUC of ROC curve, despite precision being low! Therefore it is important to look at both metrics, along with `F-beta score`

ROI on ad spend

- Same idea from prior: some cost `c` and return `r`

```
total_return = tp * r
```

```
total_spent = (tp + fp) * cost
```

```
roi = total_return / total_spent
    = (tp) / (tp + fp) * (r / cost)
    = precision * (r / cost)
```

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Model review and comparison

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

Model review

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
```

- Logistic regression: linear classifier identifying decision boundary
- Decision trees: tree-format of conditions
- Random Forests: ensemble of Decision Trees
- Neural Networks (MLPs): layers using linear combinations of features with a nonlinear activation function

Model implementation

Similarities

- Feature transformation and regularization
- Fitting via `classifier.fit(X_train, y_train)`
- Predictions via `predict_proba()` and `predict()`

Differences

- Decision Trees: `max_depth` , `min_samples_split`
- Random Forests: `n_estimators` , `oob_score`
- Logistic Regression: `fit_intercept` , `class_weight`
- Neural Networks: `hidden_layer_sizes` , `max_iter`

Model evaluation

- Key evaluation metrics:
  - Confusion matrix: `confusion_matrix(y_test, y_pred)`
  - Precision: `precision_score(y_test, y_pred)`
  - Recall: `precision_score(y_test, y_pred)`
  - F-beta score: `fbeta_score(y_test, y_pred, beta = 0.5)`
  - AUC of ROC curve: `roc_auc_score(y_test, y_score[:, 1])`

Main pros and cons of using neural networks

Pros

- Scalability with data
- Less need to do feature engineering
- More transferable across domains

Cons

- Less powerful on smaller datasets
- Difficult to interpret
- Computationally and financially cheaper

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Wrap-up video

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

Chapter 1

- Introduction to CTRs
  - Learned about basic problem from a classification lens
- Overview of machine learning models
  - Practiced with logistic regression on various datasets
- Brief overview of CTR prediction
  - Applied decision trees for CTR prediction

Chapter 2

- Basic exploratory data analysis
  - Looked at specific features and variability with CTR
- Feature engineering
  - Learned hashing and created features from existing ones
- Standardization
  - Applied standard scaling and log normalization

Chapter 3

- Applications of metric evaluation
  - Learned the business interpretations of evaluation metrics through confusion matrices and an ROI framework
- Model evaluation
  - Evaluated precision and recall relative to a baseline classifier
- Tuning models
  - Learned concepts of regularization and cross-validation
- Ensembles and hyperparameter tuning
  - Tuned hyperparameters using grid search for a Random Forest

Chapter 4

- Basic concepts and model
  - Learned the inner workings of neural networks
- Hyperparameter tuning
  - Tuned hyperparameters of neural networks via hidden layers and max iterations
- Model evaluation
  - Computed `F-beta` scores and implications of precision versus AUC of ROC curve
- Model review and comparison
  - Reviewed all models covered and compared them on all evaluation metrics

Thank you!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON