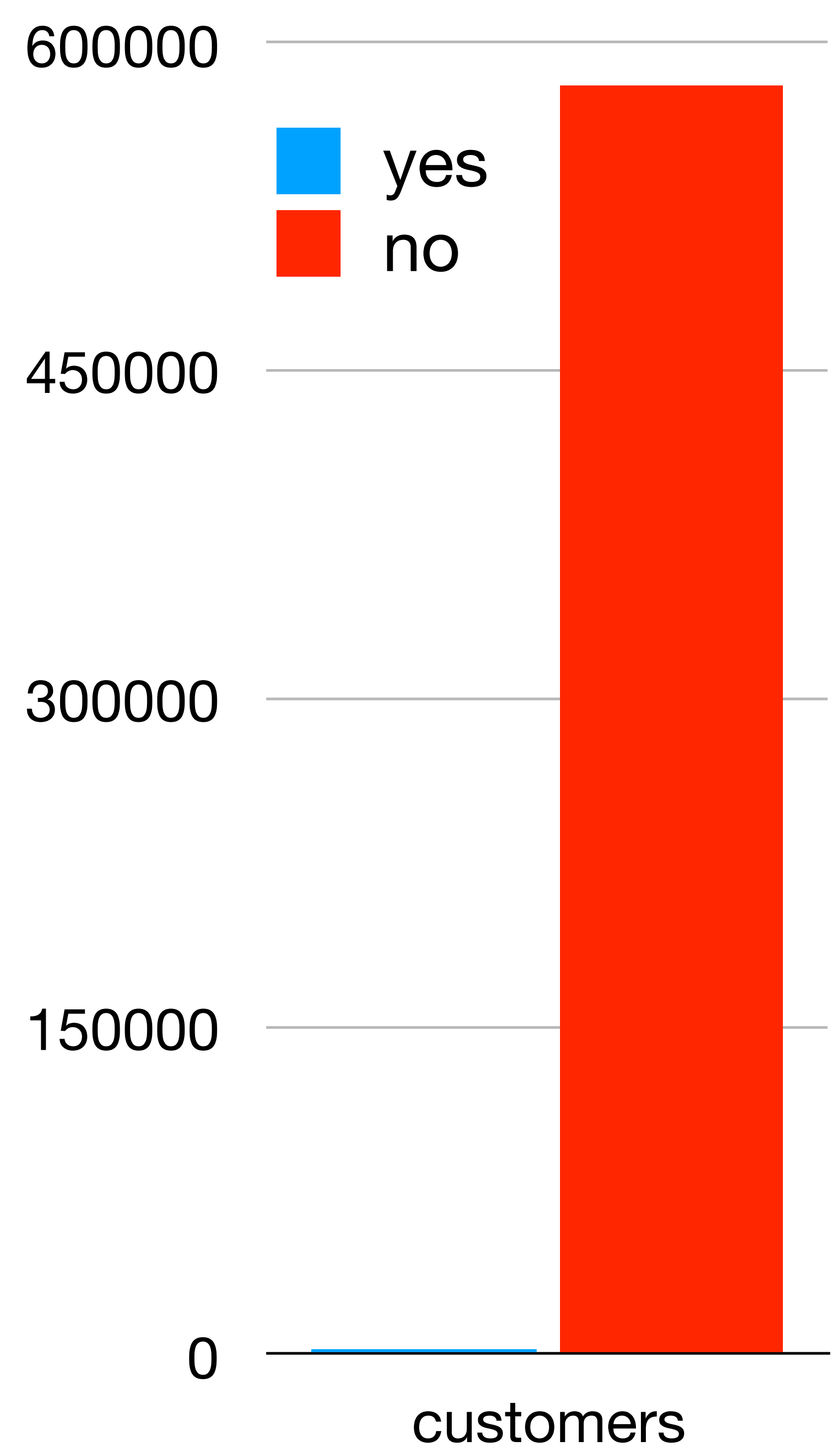
Telenor Handset Model Prediction

A Binary Classification Problem

Chentian Jiang

The Binary Classification Problem

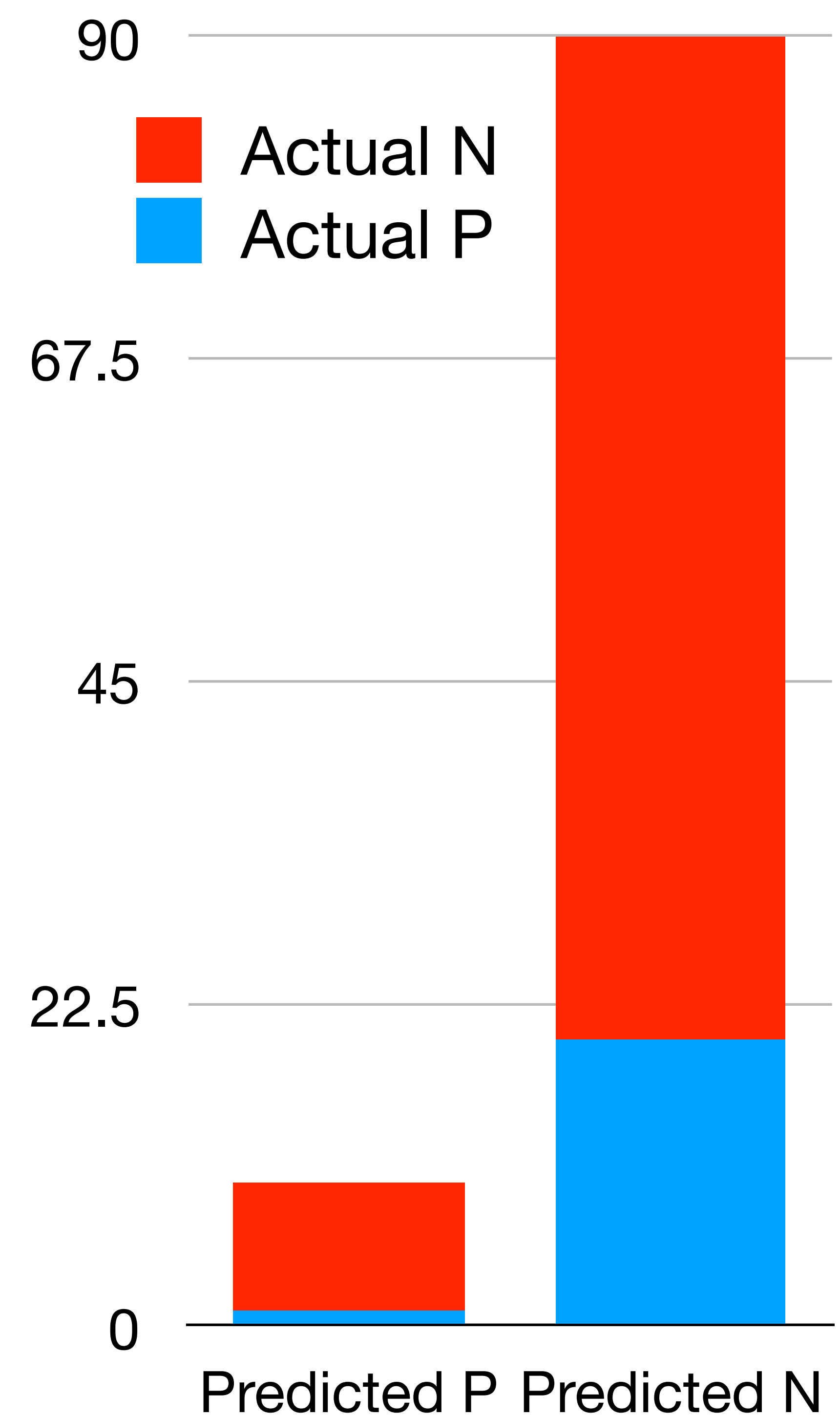
- Will the customer accept an Apple offer from Telenor?
- Predict: yes (1) or no (0)
- Dataset: 583,291 customers x 345 features, 1 binary label column
  - MPP: child
  - CU: parent who buys the subscription
- Imbalance: ~0.5% yes
  - yes: 2907
  - no: 580,384



Metrics for Unbalanced Data

		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

- Accuracy = true / all
- **Precision** = TP / (TP + FP)
    - FP: annoy customer with more offers
- Recall = TP / (TP + FN)
    - FN: lose potential profit from customer



Feature Selection

- Preprocessing
  - Encoded categorical features into -1 and 1
  - Standardized numerical features
- **f statistic** (ANOVA): ratio of two variances
  - Between each feature and the label
- Correlation between each feature and the label (>0.04)
- Split the dataset into 2 groups: data with label=-1 and data with label=1
  - **Difference in mean (>0.5)**
  - Difference in variance (>1.5)
  - Difference in median (>0.5)

Benchmark 1: Random Model

- numpy random generator: [0, 1)
  - threshold=0.5
- Tested on the whole label column
- Precision: ~**0.5**%
- Recall: ~50%
- Accuracy: ~50%

Preprocessing

- Categorical
  - one-hot
  - 1/0
- Numerical
  - standardized
  - binary features encoded as 1/0

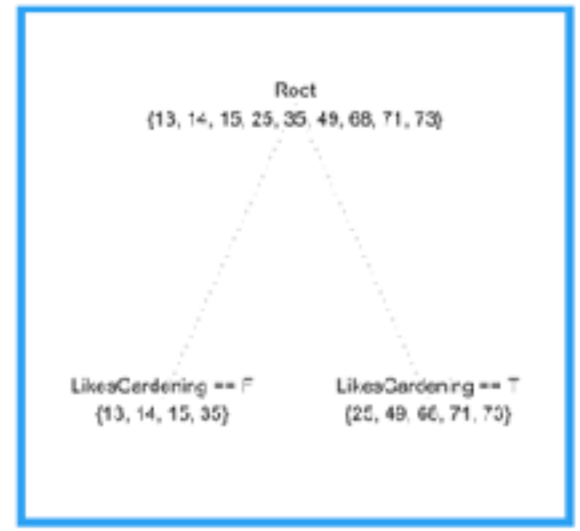
Benchmark 2: Humberto’s NN Model with TERM Features

- 10 epochs
- Results (on test data):
    - Precision: **1.72**%
    - Recall: 42.9%
    - Accuracy: 82.7%

(Main) Benchmark 3: Humberto's NN Model with F Statistic Features

- 10 epochs
- Results (on test data):
  - Precision: **1.89**%
  - Recall: 43.5%
  - Accuracy: 83.8%

Gradient Boosting: XGBoost



- Ensemble of weak learners (e.g. decision trees): each learner improves upon the previous

- Uses gradient descent to minimize a loss function

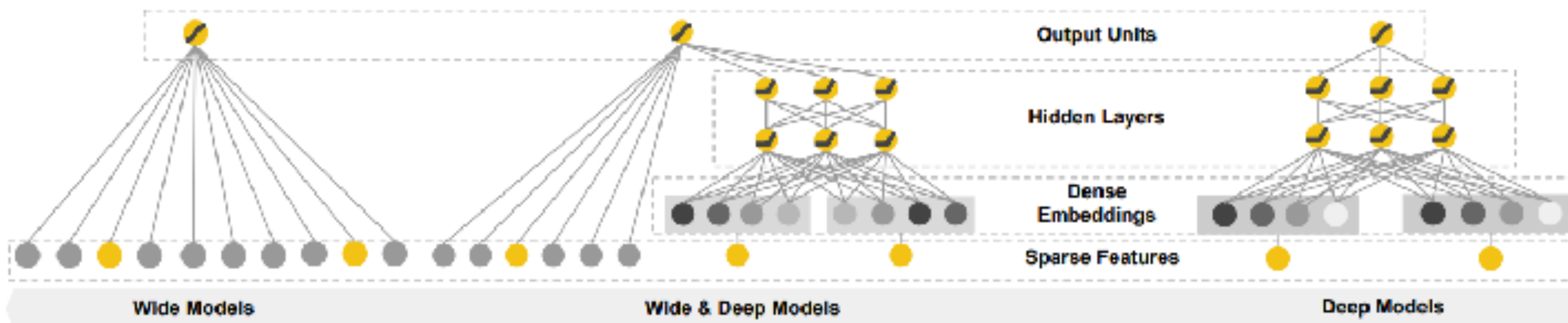
Model 1:
XGBoost with F Statistic Features

- Scale positive weight: neg examples/ pos examples
  - “oversampling”
- Results (on test data):
  - Precision: **1.49**% (lower)
  - Recall: 74.2% (higher)
  - Accuracy: 75.5% (lower)

Deep Learning Model: Starting Point

- Humberto's neural network model
  - Preprocessing:
    - Categorical: **one-hot**, integer (with embeddings), hashing
    - Numerical: **standardization**, pca whitening
    - Binary/Categorical encoding: **0/1**, -1/1
  - Feed-forward neural network with oversampling

Note: Google's Wide and Deep Learning Model



- Tensorflow tutorial: tflearn
- no **oversampling**
- 0 precision, 0 recall, 99.5% accuracy (worse than random model)

Complementary Neural Networks (CMTNN)

- Murdoch University

- Take-aways:

  - Experiment with thresholds

  - Experiment with ratio (data with label=0 to data with label=1) in the training examples

    - oversampling ratio = 3 (negative to positive)

  - Improve the model itself

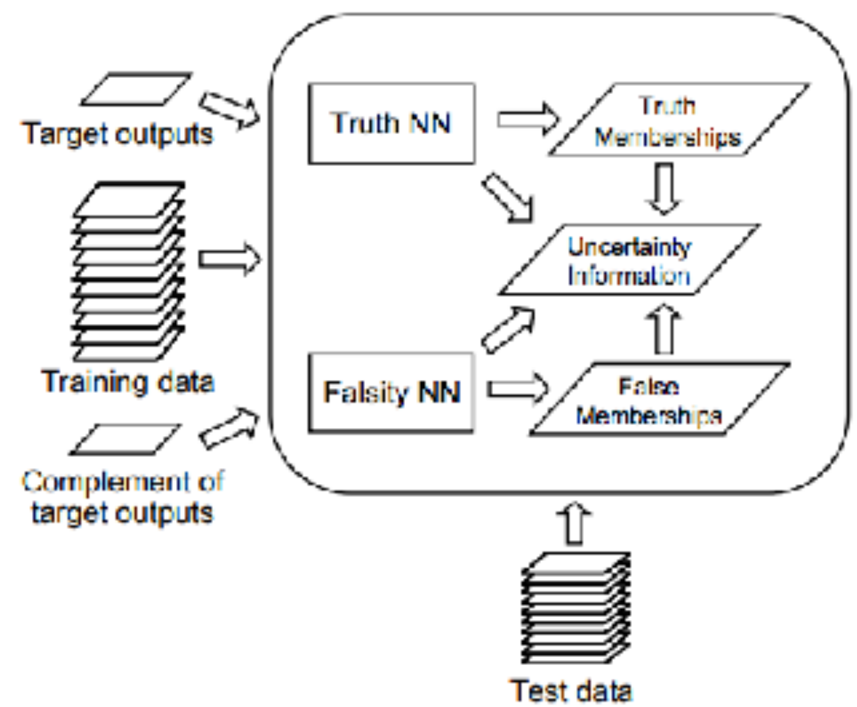


Fig. 1. Complementary neural network [11]

Model 2:
CMTNN for Humberto’s NN Model
with F Statistic Features

- Truth/Normal Model Results (on test data):
  - Precision: **1.72**% (same)
  - Recall: 44.0% (same)
  - Accuracy: 81.8% (same)
- CMTNN Model Results (on test data):
  - Precision: **1.82**% (same)
  - Recall: 57.1% (higher)
  - Accuracy: 84.5% (same)

Product-Based Neural Networks (PNN)

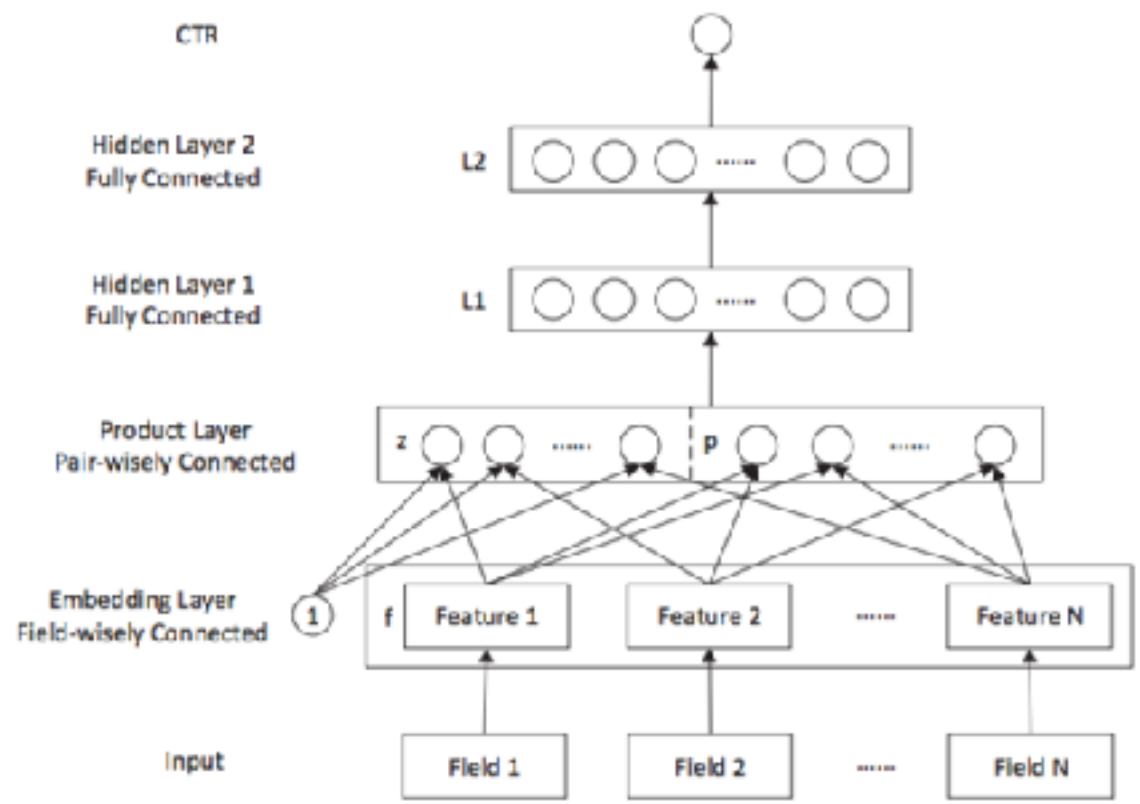


Fig. 1: Product-based Neural Network Architecture.

- Shanghai Jiao Tong University, University College London, click-through-rate
- “high-order latent patterns” in sparse categorical data
- **PNN1**, PNN2, LR, **FM**, FNN, CCPM
- higher oversampling ratio —> higher precision, higher accuracy but lower true positive rate

Model 3: PNN with All Categorical Features, Ratio=3

- Plug in Oversampling Batch Generator
- Results (on test data):
  - Precision: **2.67**% (higher)
  - Recall: 42.2% (same)
  - Accuracy: 92.1% (higher)

Model 4:
PNN with F Statistic Features (cat+num), Ratio=3

- Plug in Oversampling Batch Generator
- Numerical Model: Multilayer Perceptron
  - no dropout, no batch normalization
  - 3 hidden layers, 128 nodes each, ReLU
- Categorical Model: PNN
- Results (on test data):
  - Precision: **2.17**% (higher)
  - Recall: 41.0% (same)
  - Accuracy: 90.5% (higher)

PNN with F Statistic Features (cat+num), Ratio=3, cont.

- from epoch to epoch: model tries to find a balance between true positives and false positives
- somehow the test precision and test recall are more than x2 of their training counterparts
- not as good as PNN with all cat features
  - maybe the F Statistic Features (15) are too few? —> tried with all features I found through feature selection (not much difference)
  - or we don’t really need numerical features?

Thoughts…

- Anomaly Detection?