# COMP40270 - Cognitive Modelling

Alessio Gravina

a.y. 2018/2019

## Theory

“to identify members of a given category people compare the given instance with the examples in their memory, the most similar category is returned”

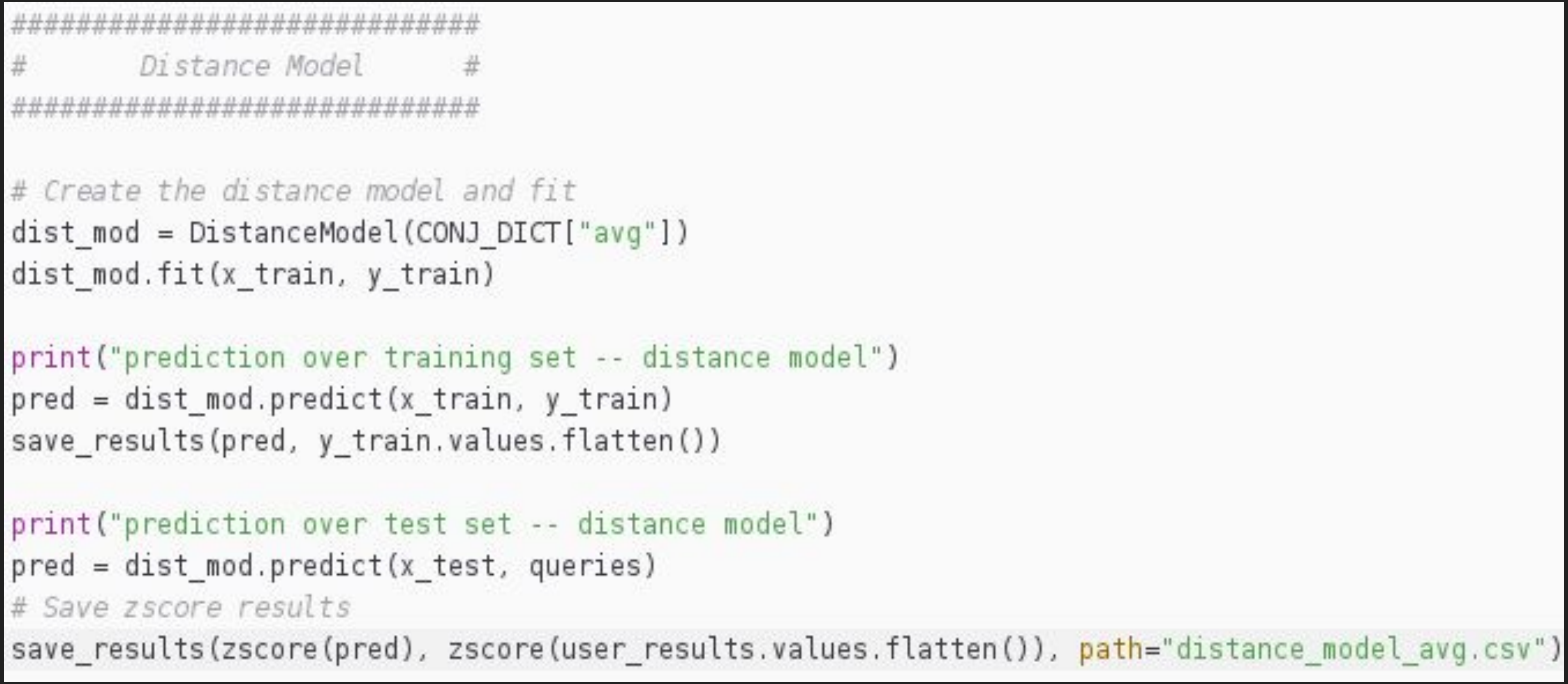
## Implementation

- `Assumption`: the model is fitted over the all possible single categories

- `Distance model`
    - given a new ***query*** instance
        - compute the distance between ***query*** and the instances in ***memory***
        - distance given by number of mismatch
    - return the membership for the category in the query

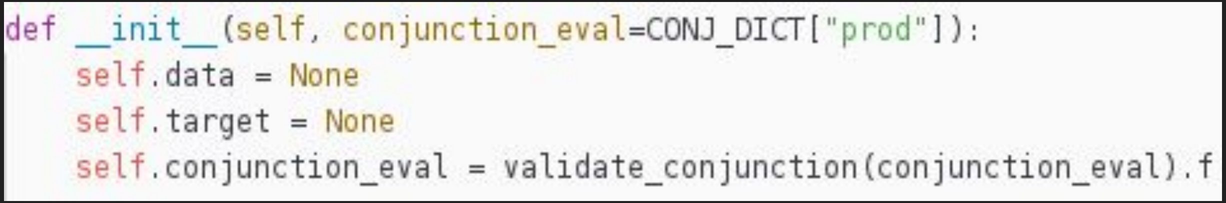
- Conjunctions not in memory → compute the ***avg*** of single categories
    - *fast* response

## Code

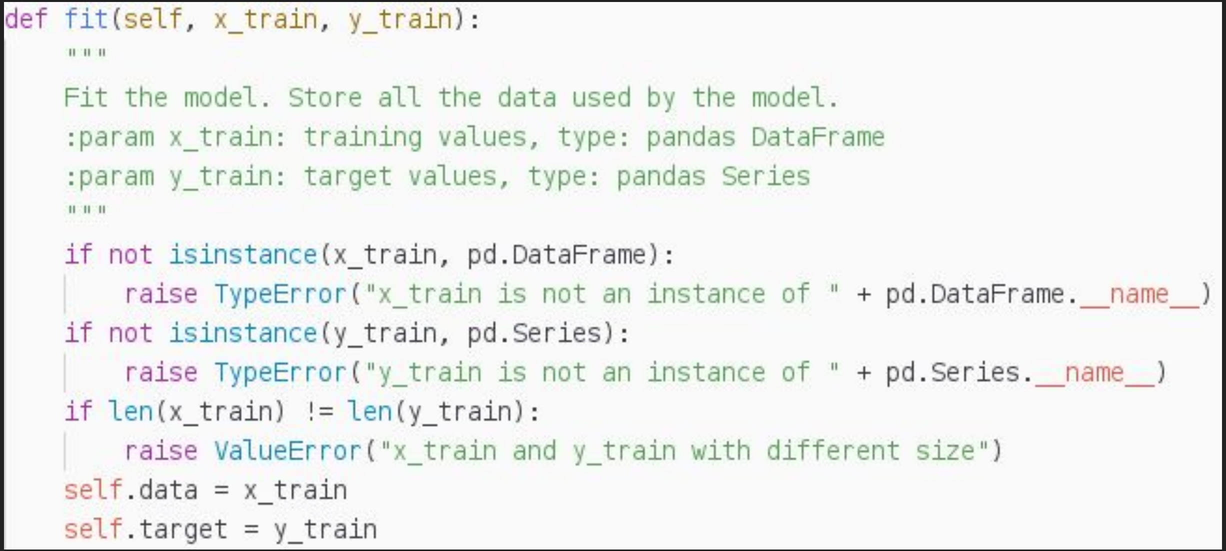


```
#############################
#       Distance Model      #
#############################

# Create the distance model and fit
dist_mod = DistanceModel(CONJ_DICT["avg"])
dist_mod.fit(x_train, y_train)

print("prediction over training set -- distance model")
pred = dist_mod.predict(x_train, y_train)
save_results(pred, y_train.values.flatten())

print("prediction over test set -- distance model")
pred = dist_mod.predict(x_test, queries)
# Save zscore results
save_results(zscore(pred), zscore(user_results.values.flatten()), path="distance_model_avg.csv")
```

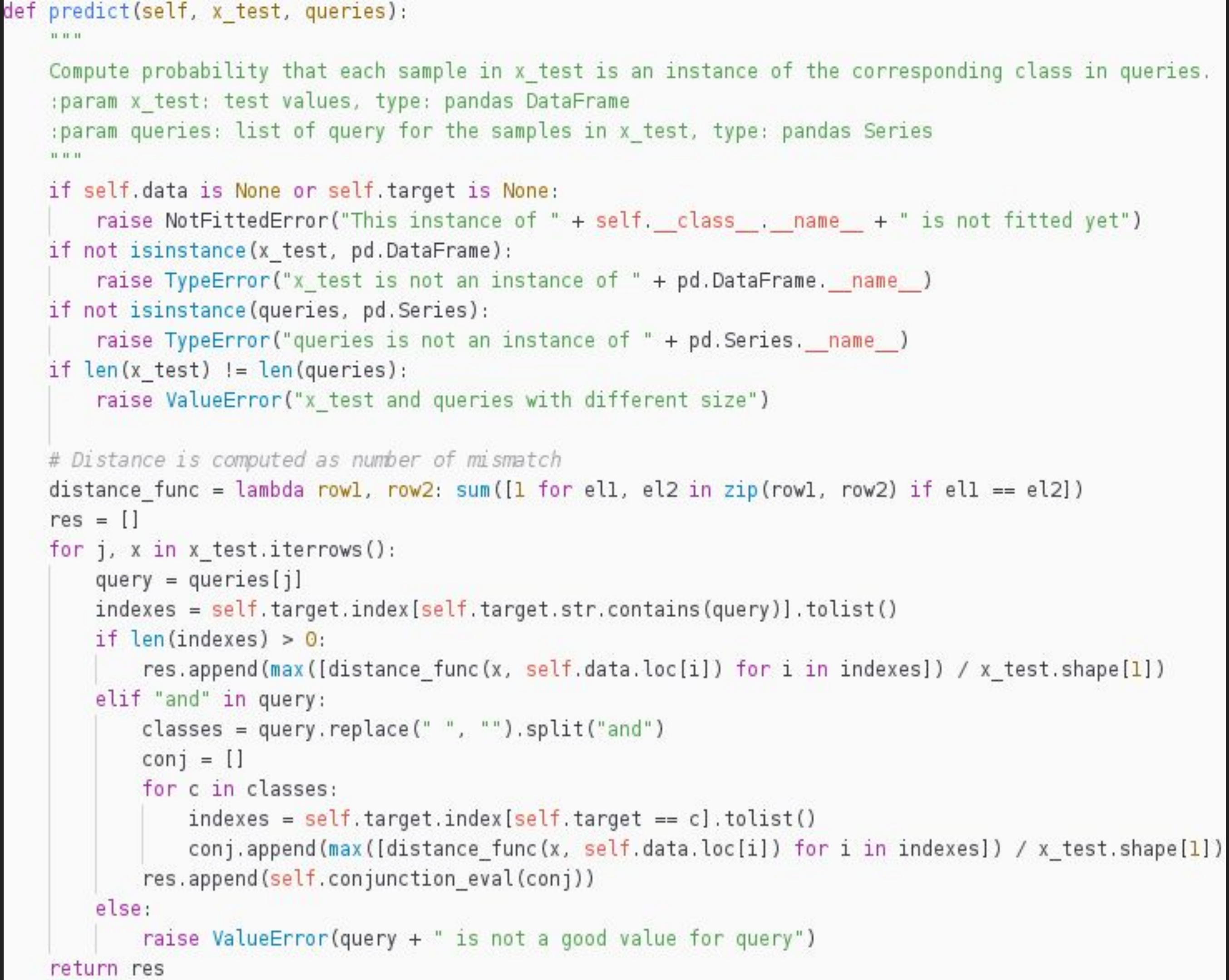
## Code - DistanceModel



```python
def __init__(self, conjunction_eval=CONJ_DICT["prod"]):
    self.data = None
    self.target = None
    self.conjunction_eval = validate_conjunction(conjunction_eval).f
```

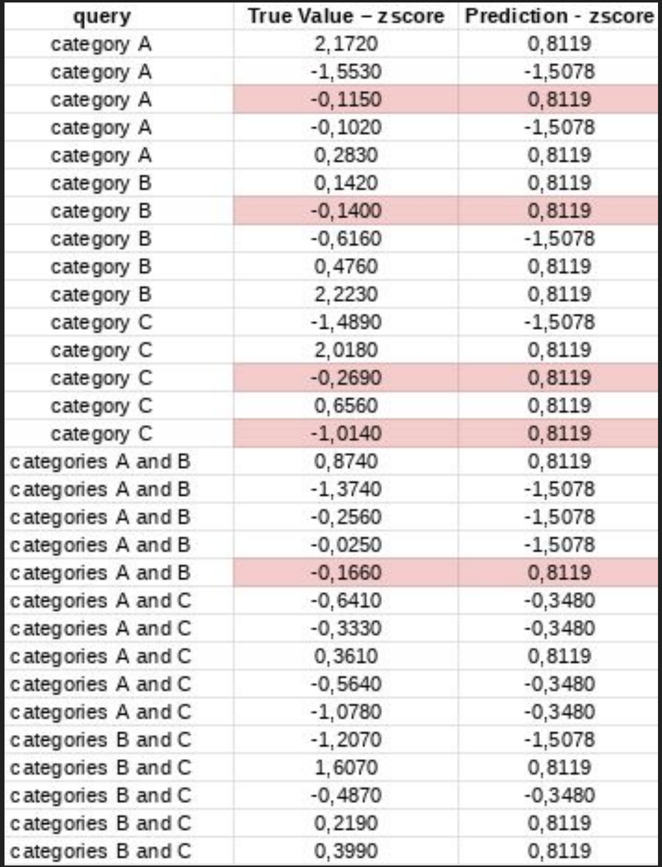


```python
def fit(self, x_train, y_train):
    """
    Fit the model. Store all the data used by the model.
    :param x_train: training values, type: pandas DataFrame
    :param y_train: target values, type: pandas Series
    """
    if not isinstance(x_train, pd.DataFrame):
        raise TypeError("x_train is not an instance of " + pd.DataFrame.__name__)
    if not isinstance(y_train, pd.Series):
        raise TypeError("y_train is not an instance of " + pd.Series.__name__)
    if len(x_train) != len(y_train):
        raise ValueError("x_train and y_train with different size")
    self.data = x_train
    self.target = y_train
```

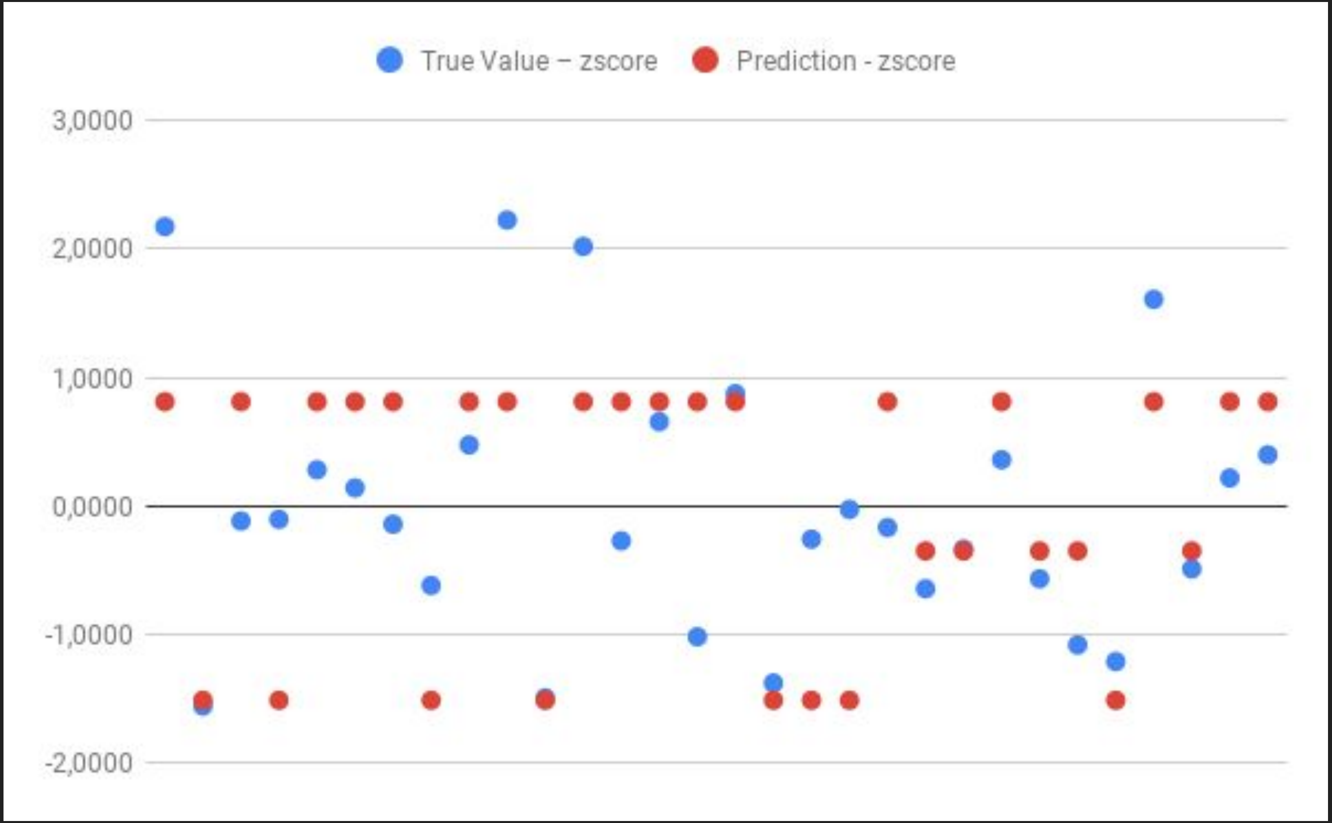


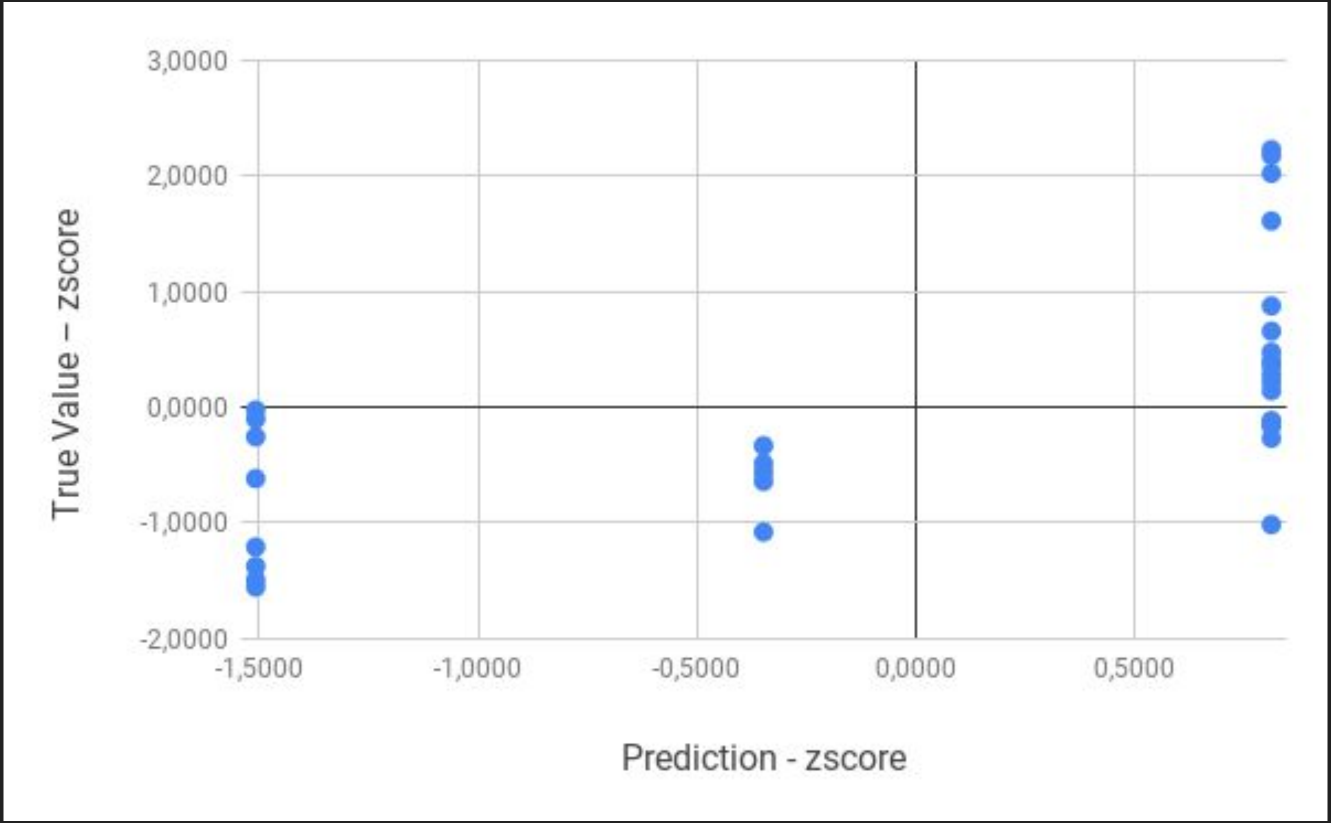
```python
def predict(self, x_test, queries):
    """
    Compute probability that each sample in x_test is an instance of the corresponding class in queries.
    :param x_test: test values, type: pandas DataFrame
    :param queries: list of query for the samples in x_test, type: pandas Series
    """
    if self.data is None or self.target is None:
        raise NotFittedError("This instance of " + self.__class__.__name__ + " is not fitted yet")
    if not isinstance(x_test, pd.DataFrame):
        raise TypeError("x_test is not an instance of " + pd.DataFrame.__name__)
    if not isinstance(queries, pd.Series):
        raise TypeError("queries is not an instance of " + pd.Series.__name__)
    if len(x_test) != len(queries):
        raise ValueError("x_test and queries with different size")

    # Distance is computed as number of mismatch
    distance_func = lambda row1, row2: sum([1 for el1, el2 in zip(row1, row2) if el1 == el2])
    res = []
    for j, x in x_test.iterrows():
        query = queries[j]
        indexes = self.target.index[self.target.str.contains(query)].tolist()
        if len(indexes) > 0:
            res.append(max([distance_func(x, self.data.loc[i]) for i in indexes]) / x_test.shape[1])
        elif "and" in query:
            classes = query.replace(" ", "").split("and")
            conj = []
            for c in classes:
                indexes = self.target.index[self.target == c].tolist()
                conj.append(max([distance_func(x, self.data.loc[i]) for i in indexes]) / x_test.shape[1])
            res.append(self.conjunction_eval(conj))
        else:
            raise ValueError(query + " is not a good value for query")
    return res
```

## Results

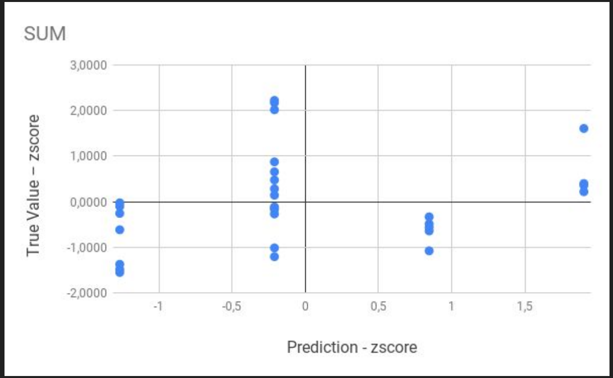


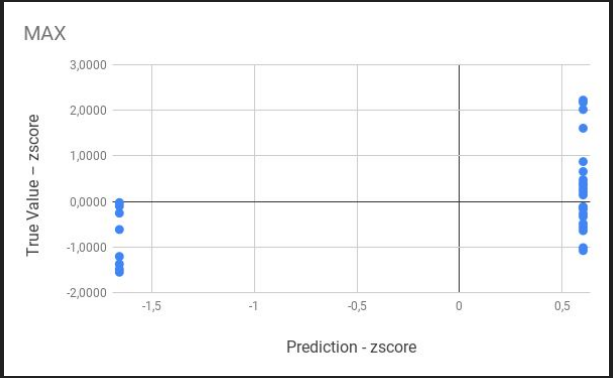
| query | True Value – zscore | Prediction - zscore |
|---|---|---|
| category A | 2,1720 | 0,8119 |
| category A | -1,5530 | -1,5078 |
| category A | -0,1150 | 0,8119 |
| category A | -0,1020 | -1,5078 |
| category A | 0,2830 | 0,8119 |
| category B | 0,1420 | 0,8119 |
| category B | -0,1400 | 0,8119 |
| category B | -0,6160 | -1,5078 |
| category B | 0,4760 | 0,8119 |
| category B | 2,2230 | 0,8119 |
| category C | -1,4890 | -1,5078 |
| category C | 2,0180 | 0,8119 |
| category C | -0,2690 | 0,8119 |
| category C | 0,6560 | 0,8119 |
| category C | -1,0140 | 0,8119 |
| categories A and B | 0,8740 | 0,8119 |
| categories A and B | -1,3740 | -1,5078 |
| categories A and B | -0,2560 | -1,5078 |
| categories A and B | -0,0250 | -1,5078 |
| categories A and B | -0,1660 | 0,8119 |
| categories A and C | -0,6410 | -0,3480 |
| categories A and C | -0,3330 | -0,3480 |
| categories A and C | 0,3610 | 0,8119 |
| categories A and C | -0,5640 | -0,3480 |
| categories A and C | -1,0780 | -0,3480 |
| categories B and C | -1,2070 | -1,5078 |
| categories B and C | 1,6070 | 0,8119 |
| categories B and C | -0,4870 | -0,3480 |
| categories B and C | 0,2190 | 0,8119 |
| categories B and C | 0,3990 | 0,8119 |

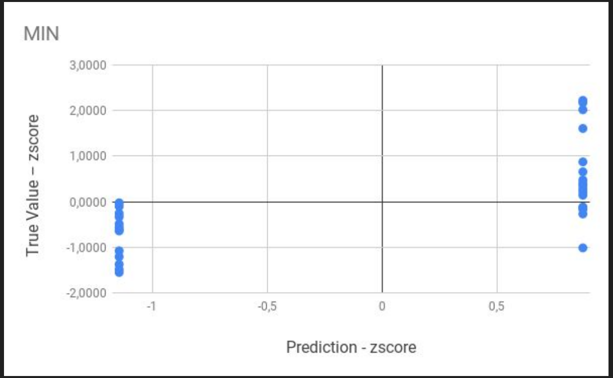

● True Value – zscore
● Prediction - zscore
3,0000
2,0000
1,0000
0,0000
-1,0000
-2,0000

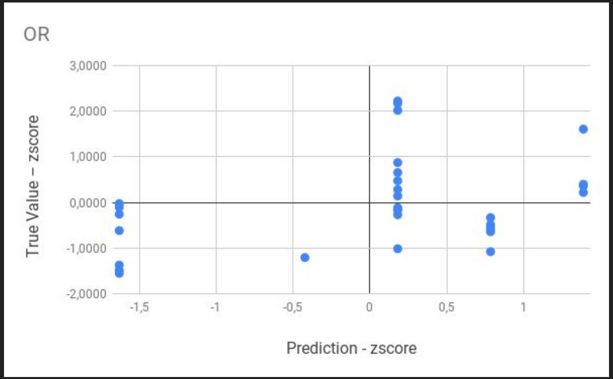

3,0000
2,0000
1,0000
0,0000
-1,0000
-2,0000
True Value – zscore
-1,5000
-1,0000
-0,5000
0,0000
0,5000
Prediction - zscore

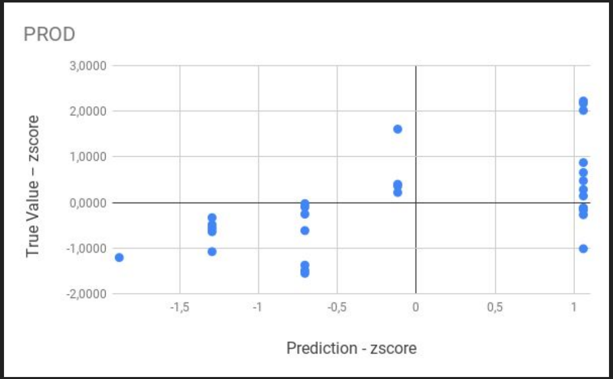
## Other experiments











## Conclusion

- Distance model
  - not economical
  - expensive in memory and time
    - compute similarity over all examples
    - store all the training set

- avg for a quick judgment
  - irrational approach

- In general the model
  - works well with conjunctions
  - some error with single categories