Perceptron

Learning Portfolio 3

universität innsbruck

Data Set

The data set is a data set retrieved on kaggle. It contains data about Titanic passengers and whether they survived the crash.

The goal of the data set is to predict whether the passengers died or not.

```
[3]  data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

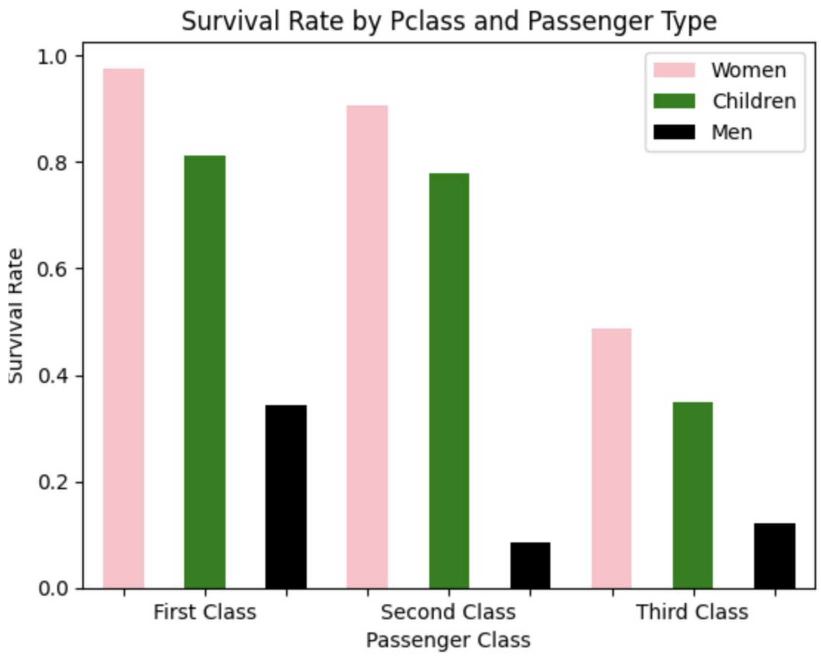
Data Preprocessing

- Missing values in “Age” -> Imputed by clustering
- Missing values in “Embarked” -> Imputed by Mode
- New Feature “Passenger Type” -> Child, Women or Men (ordinal)
- “SibSp”, “Parch” -> Dichotomous reduction
- Logarithmic transformation and scaling
- One-Hot-Encoding for categorical data

```python
# drop not needed data
data = data.drop("Cabin", axis=1)
data = data.drop("Ticket", axis=1)
data = data.drop("Name", axis=1)

# impute passenger age using k-nearest neighbors
from sklearn.impute import KNNImputer

imputer = KNNImputer()
data_num = data.select_dtypes(include=[np.number])
imputer.fit(data_num)
data_new = imputer.transform(data_num)
data["Age"] = data_new[:,3]

# impute embarkation point by mode
from sklearn.impute import SimpleImputer

s_imputer = SimpleImputer(strategy="most_frequent")
s_imputer.fit(data)
data_new = s_imputer.transform(data)
data["Embarked"] = data_new[:,3]

# reduce to boolean attribute
data['SibSp'] = data['SibSp'].apply(lambda x: 0 if x == 0 else 1)
data['Parch'] = data['Parch'].apply(lambda x: 0 if x == 0 else 1)
```

Data Visualization

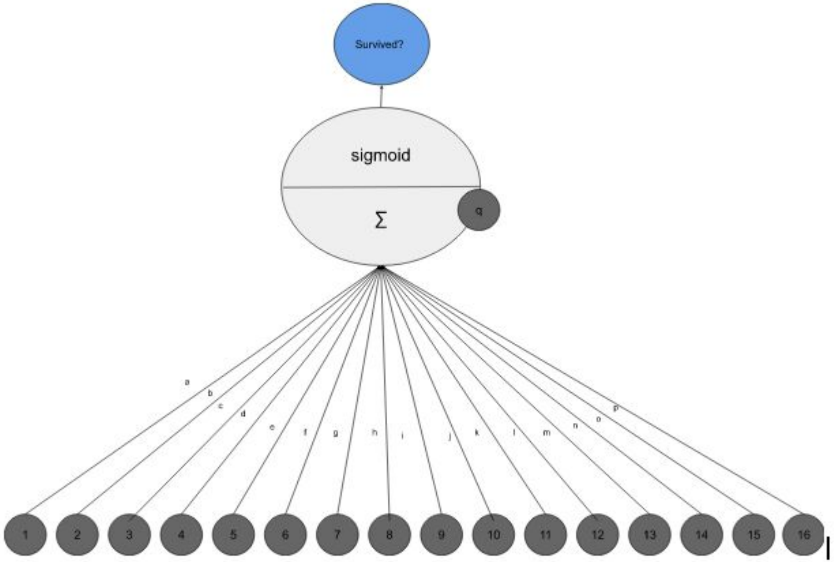
- Women were more likely to survive than children and men
- Thir passenger class was the least likely to survive



universität innsbruck

Training

- 15 weights (a to p)
- Bias q
- Activation function sigmoid



Training

- Loss function: RMSE
- Learning rate: 1
- Training until loss = previous_loss
- Local Optimum: 0.369
- 18.29 % wrong predictions on the validation data set

```python
# Initializing learning rate
lr = 1e0

def apply_step(params):
    # Generating predictions based on input array x, current param set and fur
    preds = f(tensor_data, params)
    # Calculating loss function for predictions and target values
    loss = rmse(preds, tensor_label)
    # Calculate gradient for current param set
    loss.backward()
    # Change param set according to learning rate and calculated gradients
    params.data -= lr * params.grad.data
    # Reset gradients for next iteration
    params.grad = None
    # print current loss
    print("loss ")
    print(loss.item())
    return loss.item()

# Initializing parameters for prediction function to seventeen random values
params = torch.randn(17).requires_grad_()
prev_loss = 0
# repeat until local optimum is found
while True:
  loss = apply_step(params)
  if loss == prev_loss:
    print("Local Optimum found.")
    break
  else:
    prev_loss = loss
```

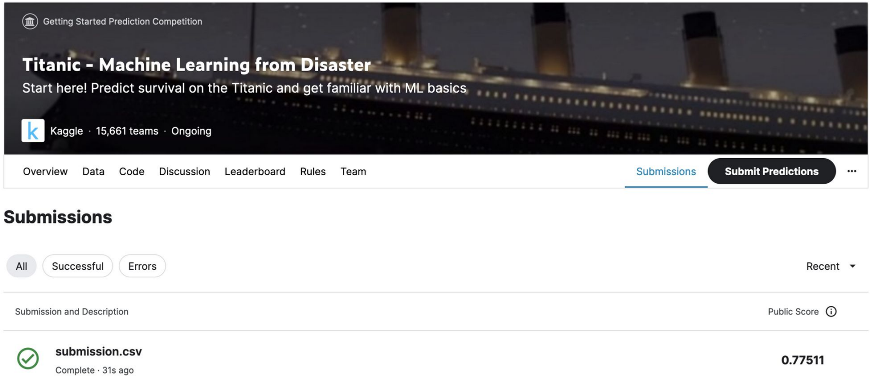
```python
[19] preds = f(tensor_data, params)
     incorrect = ((preds.round() - tensor_label).abs()).sum()
     incorrect.item()/tensor_data.shape[0]
```

```
0.1829405162738496
```

universität innsbruck

Testing

- 77.511 % accuracy
- Above average accuracy for models in kaggle



Kontakt

Fabian Leuk
12215478
fabian.leuk@student.uibk.ac.at