Artificial Neural Network

Learning Portfolio 6

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
```

universität innsbruck

Data Preprocessing

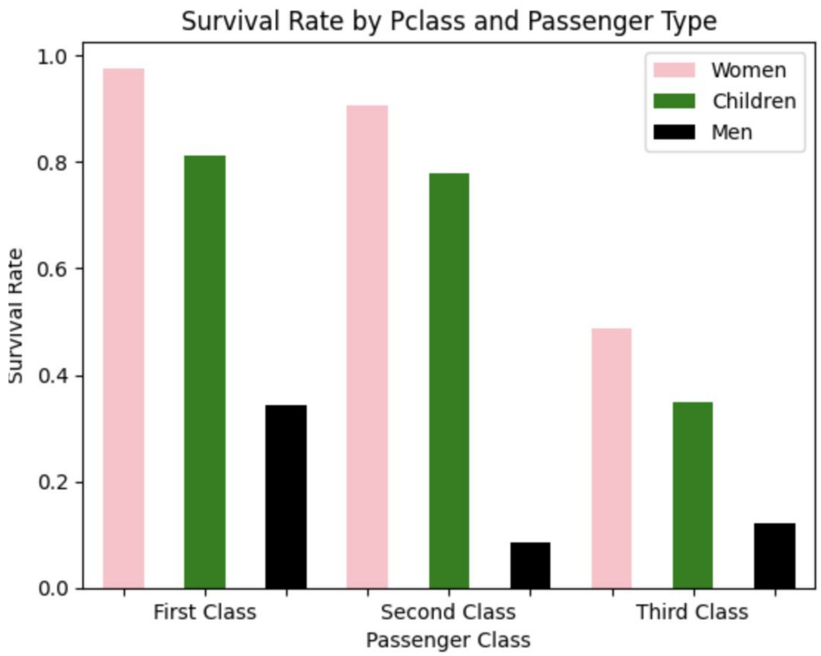
- Missing values in “Age” -> Imputed by clustering
- Missing values in “Embarked” -> Imputed by Mode
- New Feature “Passenger Type” -> Child, Women or Men (ordinal)
- New Feature “Deck” derived from Cabin
- New feature “TicketFreq”
- New feature “Alone”
- New feature “Title” derived from Name
- Logarithmic transformation and scaling
- One-Hot-Encoding for categorical data

```python
# define function to apply to each row
def passenger_type(row):
    if row['Sex'] == 'female' and row['Age'] >= 18:
        return 0
    elif row['Age'] <= 18:
        return 1
    elif row['Sex'] == 'male' and row['Age'] >= 18:
        return 2
    else:
        return None

def add_features(df):
    df['LogFare'] = np.log1p(df['Fare'])
    df['Deck'] = df.Cabin.str[0].map(dict(A="ABC", B="ABC", C="ABC", D="DE", E="DE", F="FG", G="FG"))
    df['Family'] = df.SibSp+df.Parch
    df['Alone'] = df.Family==0
    df['TicketFreq'] = df.groupby('Ticket')['Ticket'].transform('count')
    df['Title'] = df.Name.str.split(', ', expand=True)[1].str.split('.', expand=True)[0]
    df['Title'] = df.Title.map(dict(Mr="Mr",Miss="Miss",Mrs="Mrs",Master="Master"))
    df['Passenger Type'] = df.apply(passenger_type, axis=1)

add_features(data)

def impute(df):
  # impute passenger age using k-nearest neighbors
  from sklearn.impute import KNNImputer

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

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

  s_imputer = SimpleImputer(strategy="most_frequent")
  s_imputer.fit(df)
  data_new = s_imputer.transform(df)
  df["Embarked"] = data_new[:,3]
  df["Title"] = data_new[:,3]
  df["Passenger Type"] = data_new[:,3]


impute(data)
```

universität innsbruck

Data Visualization

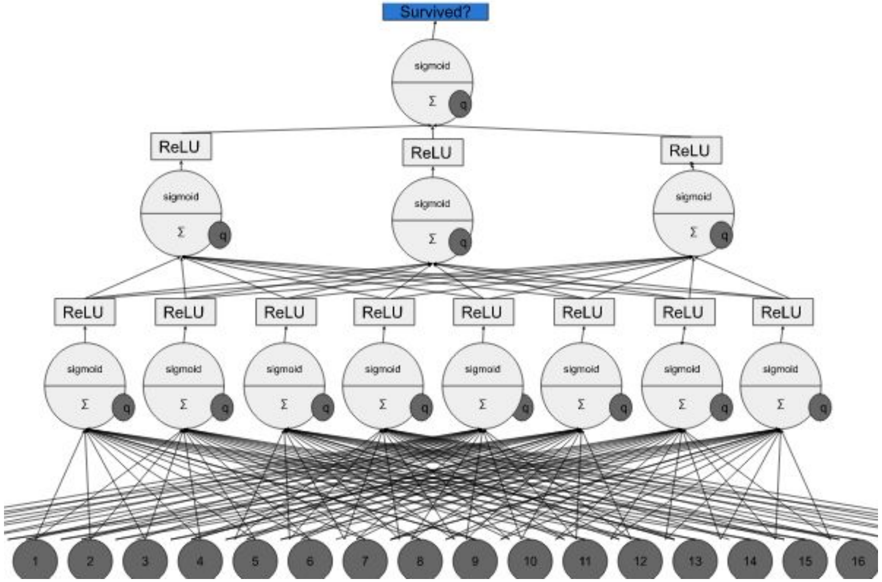
- Women were more likely to survive than children and men
- Thir passenger class was the least likely to survive



universität innsbruck

Training

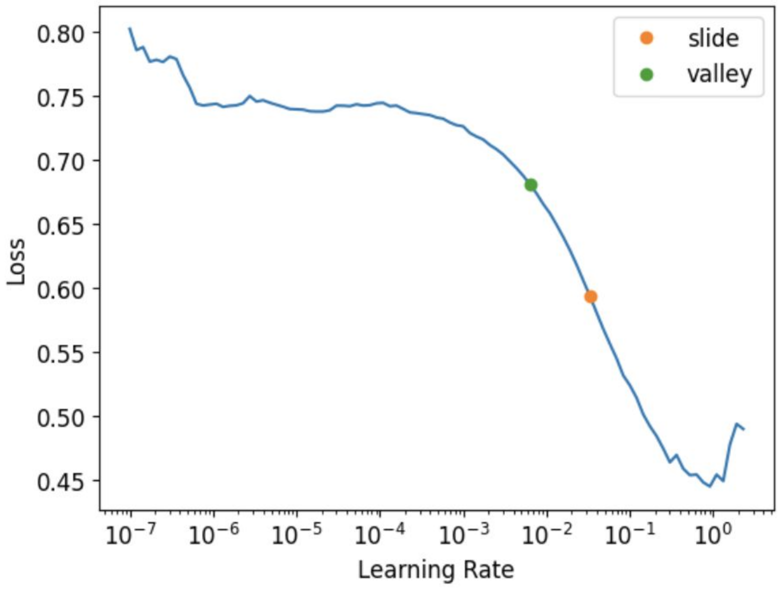
- Three-Layer-Architecture
- 10 first-layer-perceptrons
- 10 second-layer-perceptrons
- output layer
- Rectified Linear Units as elements of Non-Linearity



Training

- Learning rate was determined by calling lr_find() => 0.1
- Loss function: default
- Training for 32 epochs
- Validation metric: accuracy
- 78 % accuracy on validation data set

SuggestedLRs(slide=0.033113110810518265, valley=0.0063095735386013985)



Training

```
from fastai.tabular.all import *

splits = RandomSplitter(seed=42)(data)

dls = TabularPandas(
    data, splits=splits,
    procs = [Categorify, FillMissing, Normalize],
    cat_names=["Sex","Pclass","Embarked","Deck", "Title", "Passenger Type"],
    cont_names=['Age', 'SibSp', 'Parch', 'LogFare', 'Alone', 'TicketFreq', 'Family'],
    y_names="Survived", y_block = CategoryBlock(),
).dataloaders(path=".")
learn = tabular_learner(dls, metrics=accuracy, layers=[10,10])
learn.lr_find(suggest_funcs=(slide, valley))
```

```
[23] learn.fit(32, lr=0.1)
```

10	0.023527	1.033304	0.786517	00:00
11	0.018493	1.033936	0.786517	00:00
12	0.014601	1.017279	0.792135	00:00
13	0.011562	1.023011	0.786517	00:00
14	0.009183	1.026037	0.786517	00:00
15	0.007304	1.022543	0.792135	00:00
16	0.005859	1.032959	0.786517	00:00
17	0.004678	1.053733	0.786517	00:00
18	0.003735	1.048554	0.786517	00:00
19	0.002987	1.038666	0.786517	00:00
20	0.002394	1.041331	0.780899	00:00
21	0.001918	1.035710	0.780899	00:00
22	0.001537	1.038534	0.780899	00:00
23	0.001238	1.026991	0.786517	00:00
24	0.000995	1.033771	0.792135	00:00
25	0.000800	1.036949	0.792135	00:00
26	0.000644	1.042281	0.792135	00:00
27	0.000518	1.042844	0.797753	00:00
28	0.000421	1.048046	0.780899	00:00
29	0.000343	1.052475	0.786517	00:00
30	0.000280	1.057380	0.797753	00:00
31	0.000229	1.062997	0.786517	00:00

Testing

- 72 % accuracy single model
- 73.2 % accuracy model ensemble (8 models)
- Worse than single layer network from Learning Portfolio 3 (77.511 %)
- Different data preparation => Collinearity?
- Average accuracy for models in kaggle

single_preds.csv Complete · now · Single Learner result	**0.72009**
ensemble_preds.csv Complete · 3m ago · Ensemble prediction advanced feature engineering less overfitting	**0.73205**

universität innsbruck

Kontakt

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