06/13/2022

German E. Baltazar Reyes

THE HUNGER GAMES

CONTEXT

Several people fight with each other until one person survives.

People outside the event could place bets on who would survive.

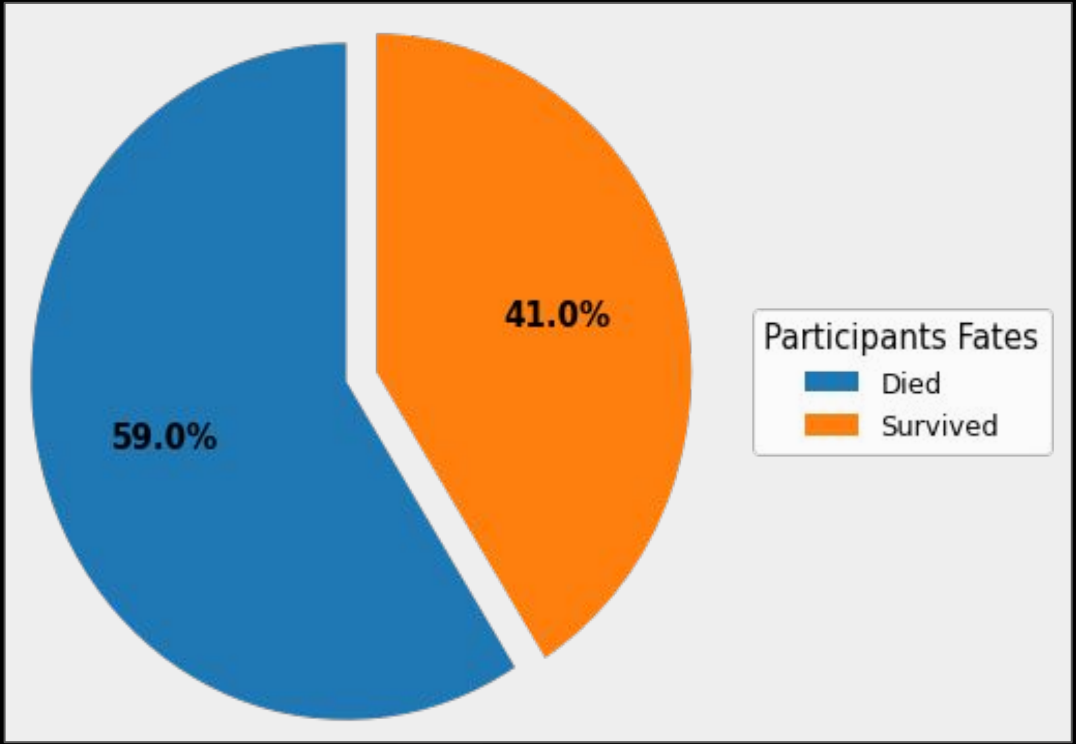
DATASET

- Titanic dataset
  - 891 observations
  - 69 features
- Gives passengers information and if they survived the accident or not
- Removed 96 repeated observations
- Selected only 11 features
  - 9 original
  - 2 engineered (fam_size and classes)

Sex (boolean)	Age (int)	Survived (boolean)
Pclass_1 (boolean)	Pclass_2 (boolean)	Pclass_3 (boolean)
Fare (float)	Small family (boolean)	Large family (boolean)

JUSTIFICATION

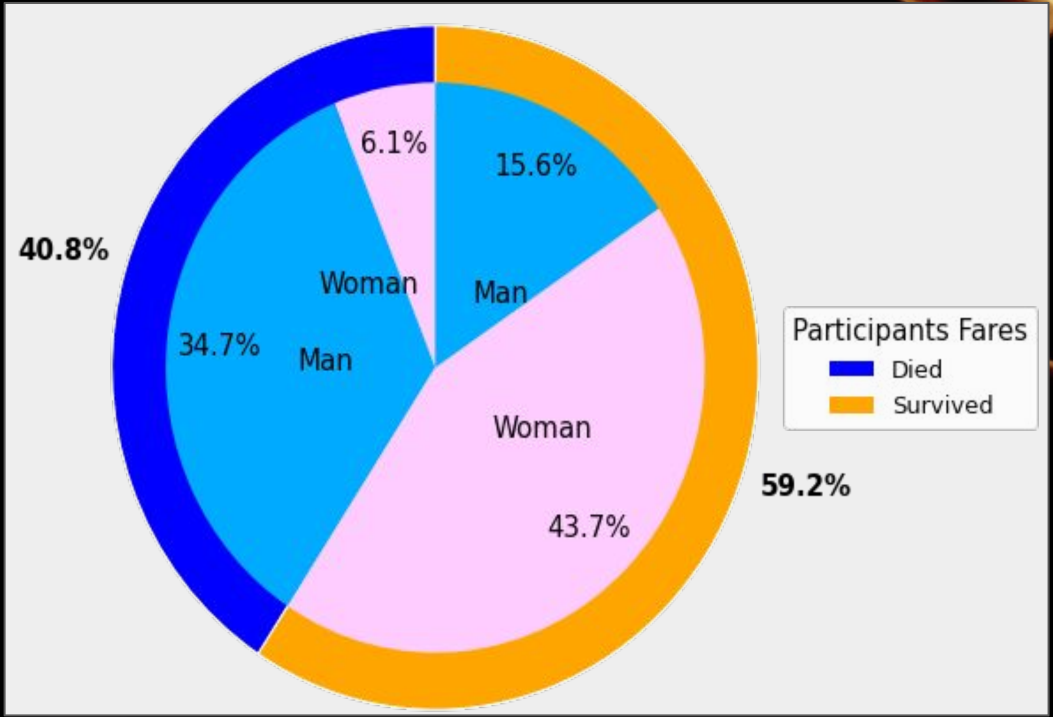
DIFFICULT TO GUESS THE SURVIVORS

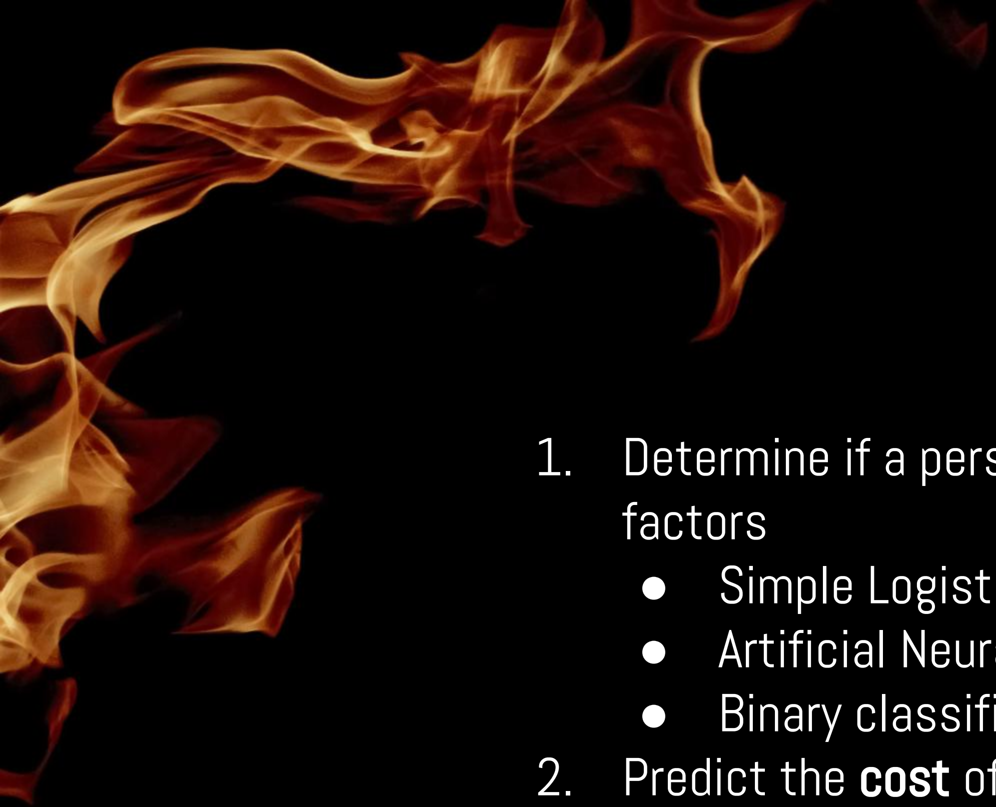


JUSTIFICATION

BARELY MORE THAN HALF OF THE MONEY IS INVESTED ON SURVIVORS

- $16,227.38 bet on survivors
  - $11,962.57 women
  - $4,264.81 men
- $11,173.85 bet on dead people
  - $1,674.94 women
  - $9,498.91 men





OBJECTIVES

1. Determine if a person can **survive** based on several factors
    - Simple Logistic Regression
    - Artificial Neural Networks
    - Binary classification
2. Predict the **cost** of its fare
    - Artificial Neural Networks
    - Regression

CLASSIFICATION TASK

DATA

- Inputs: Sex, Age, Class, and Family Size
- Output: Survived
- 70/10/15 data distribution
  - Train set: 556
  - Validation set: 120
  - Test set: 119

LOGISTIC REGRESSION

1. Train model
   a. Finetune epochs
2. Validate model
   a. Finetune learning rate
3. Test model
   a. Use best model for accuracy

ANN

1. Train model
   a. Select architecture and activations
2. Validate model
   a. Finetune learning rate and epochs
3. Test model
   a. Use best model for accuracy

REGRESSION TASK

DATA

- Inputs: Sex, Age, Class, Family Size, and Survived
- Output: Fare
- 70/10/15 data distribution
  - Train set: 556
  - Validation set: 120
  - Test set: 119

ANN

1. Train model
   a. Select architecture and activations
2. Validate model
   a. Finetune learning rate and epochs
3. Test model
   a. Use best model for accuracy

CLASSIFICATION RESULTS

	Training Accuracy	Validation Accuracy	Test Accuracy
Logistic Regression	79.67%	79.16%	76.47%
ANN	60.25%	54.16%	57.98%

REGRESSION RESULTS

	Training R^2	Validation R^2	Test R^2
ANN	0.001554	-0.00082	-0.00386

CONCLUSIONS

1. MORE DATA IS REQUIRED
2. ANN IS TO COMPLEX FOR SUCH LITTLE DATA
3. CLASSIFICATION IS AN EASIER TASK
4. PREDICTION OF FARES IS NON-LINEAR
5. DIFFICULT TO PREDICT WITH LITTLE INFORMATION

THANK YOU

Any questions?

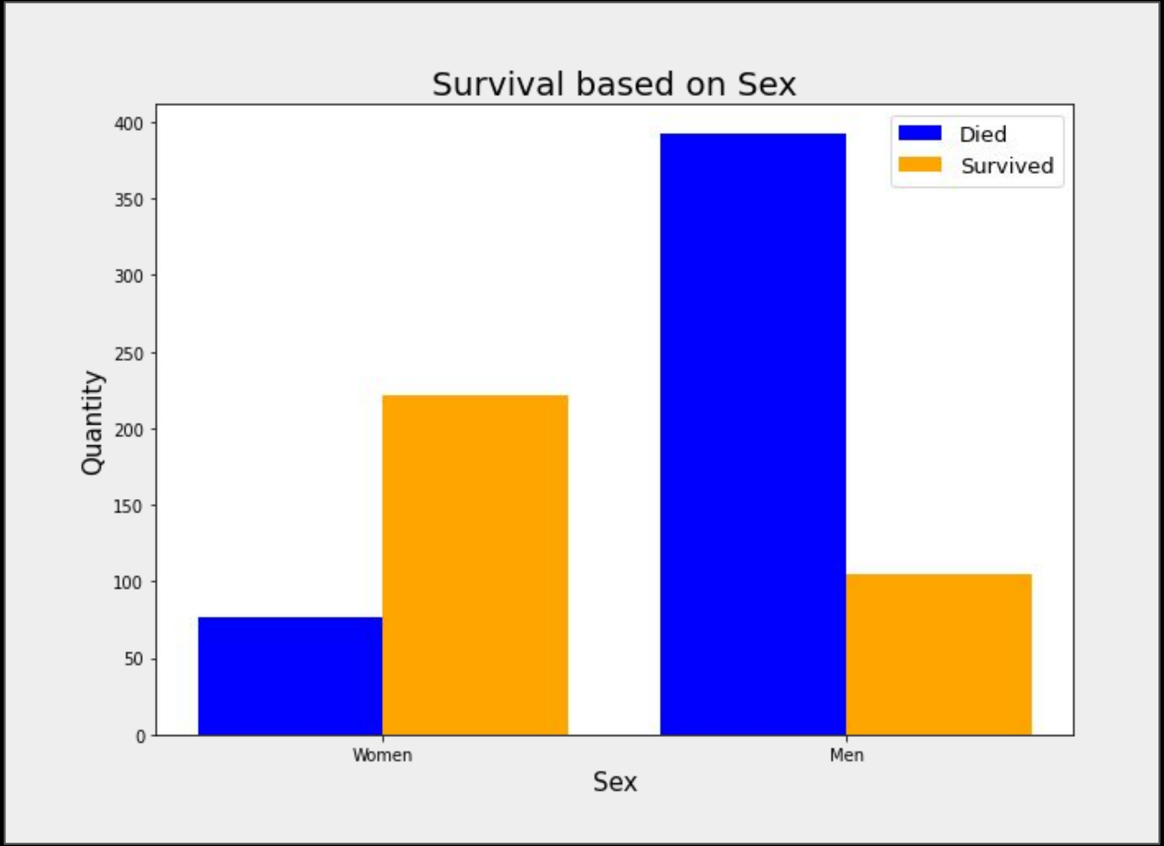
APPENDIX

REAL-WORLD USE CASE

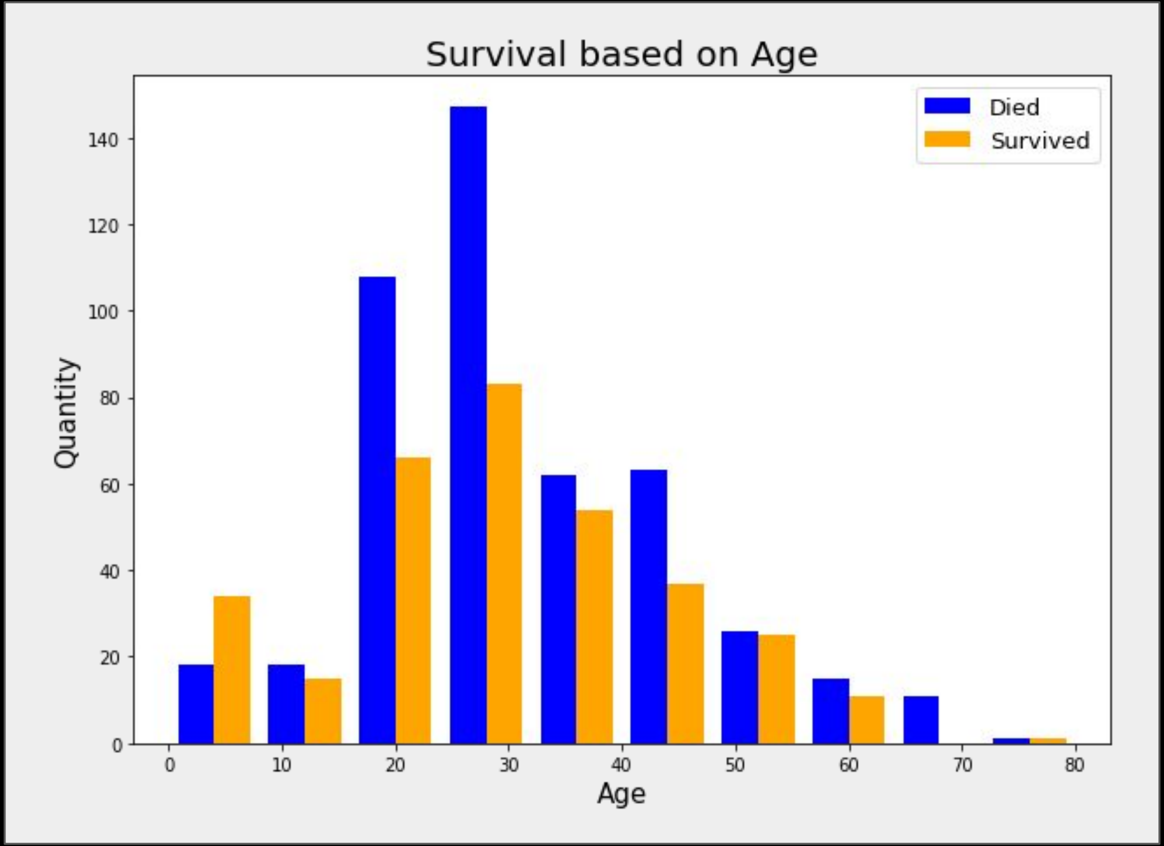
Risk analysis for tourists and people in general, so healthcare programs could determine how prone a person is to die in a particular case (boarding a ship in this case).

For the person, he can determine how much money should receive for that particular coverage.

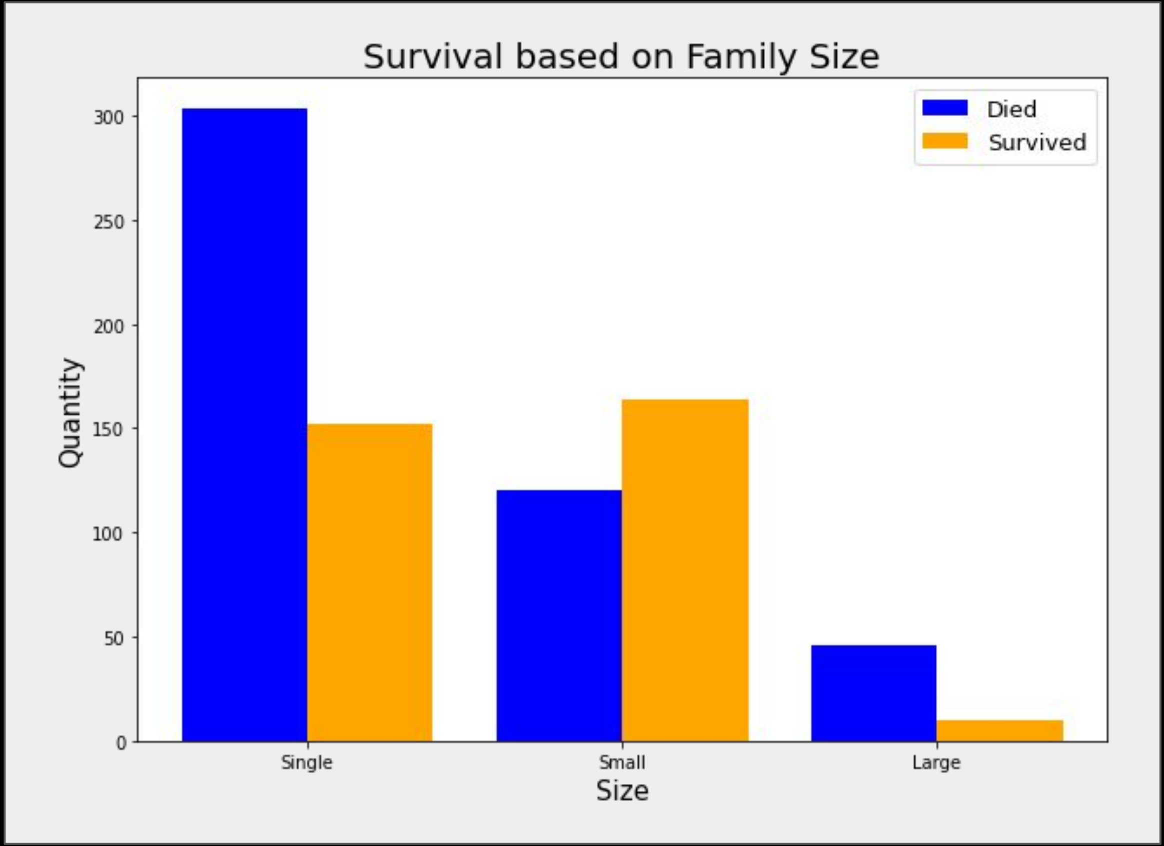
SURVIVAL ANALYSIS



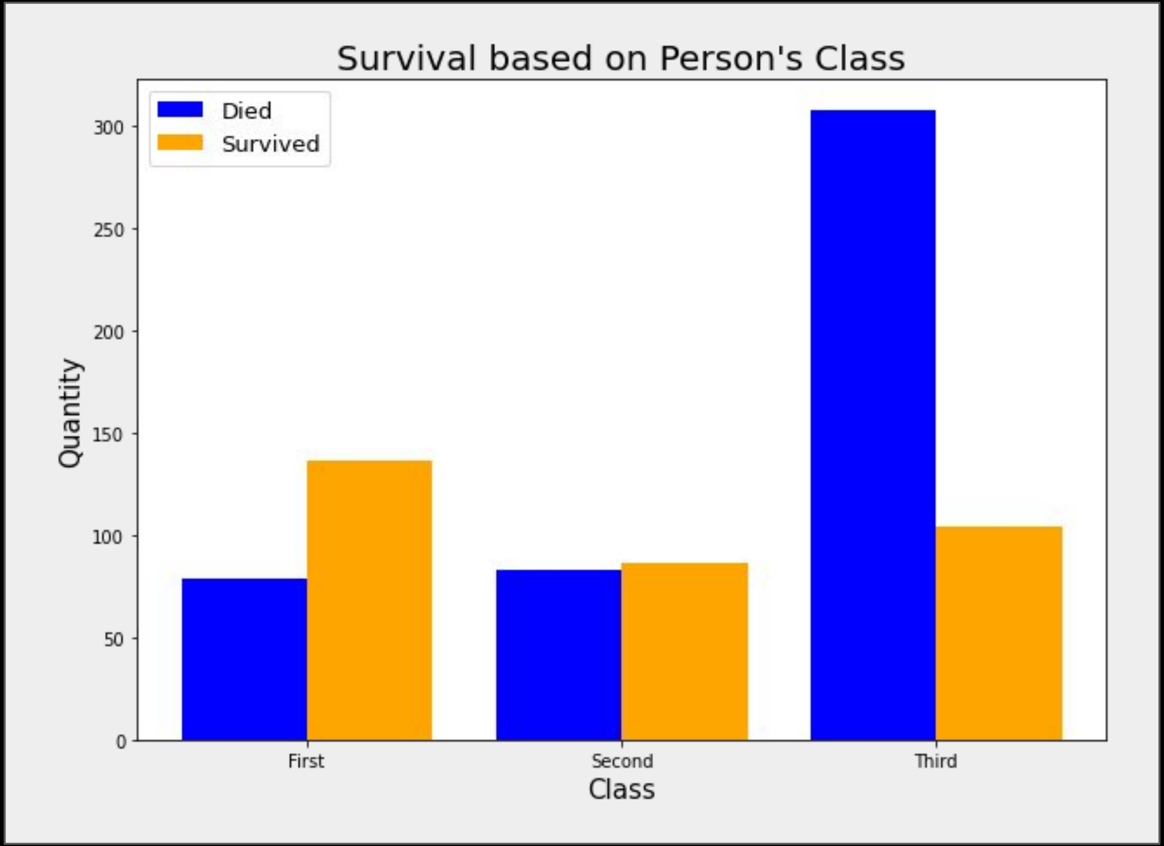
SURVIVAL ANALYSIS



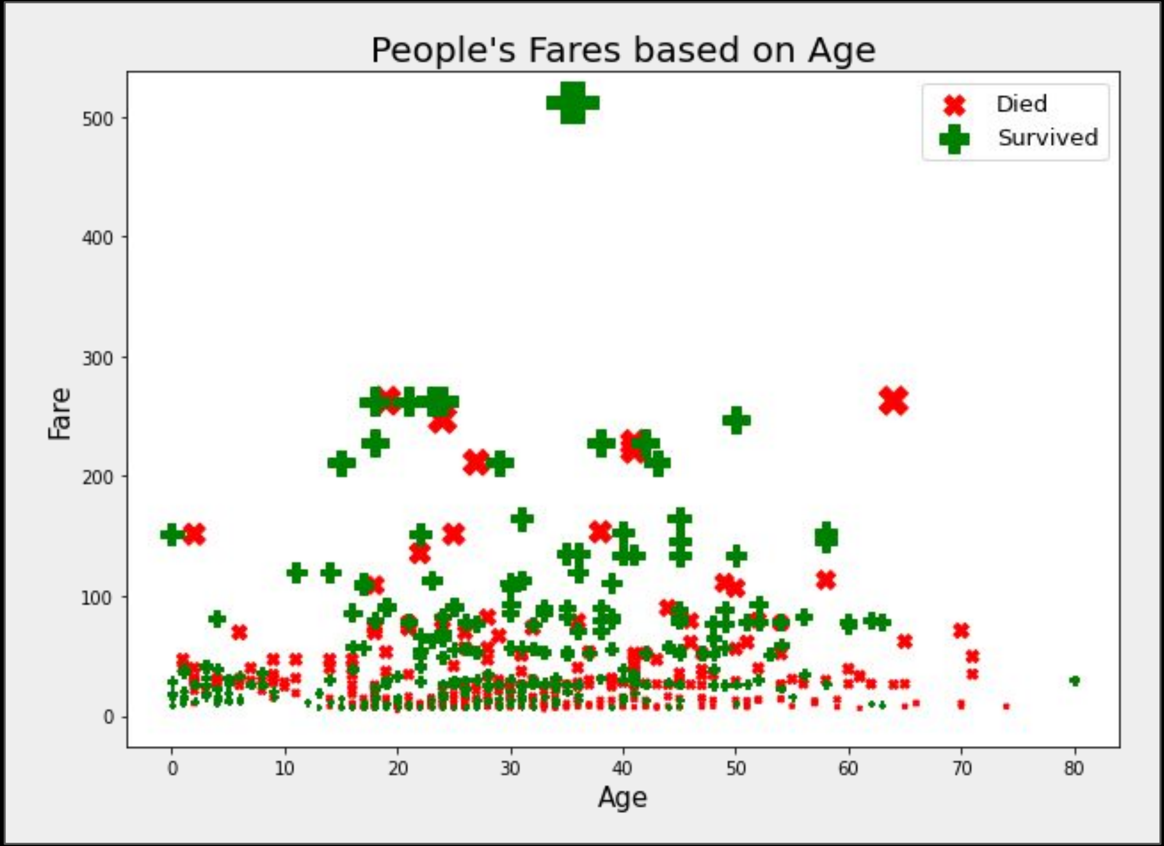
SURVIVAL ANALYSIS



Survival Analysis



FARE ANALYSIS



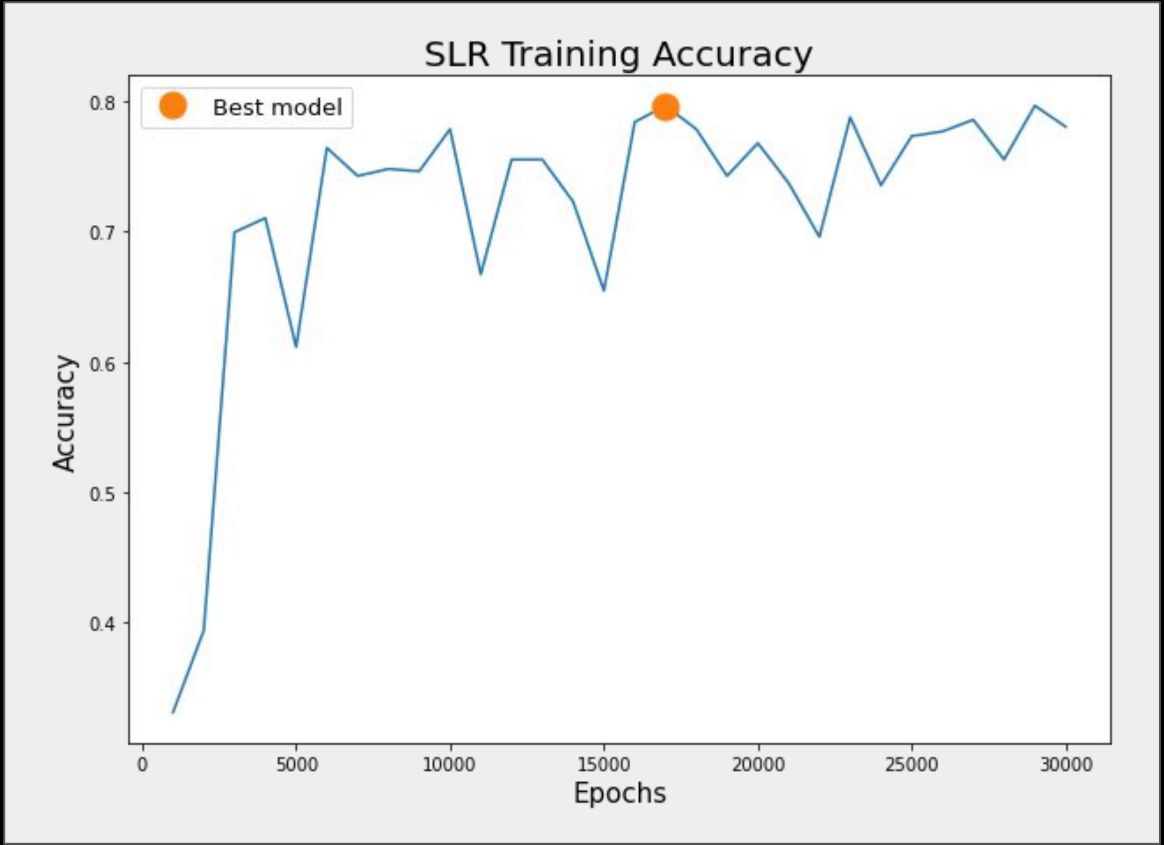
ENGINEERED FEATURES

	Small Family	**Large Family**	**Class**
Single	No	No	0
Small Family	Yes	No	1
Large Family	No	Yes	2

Ship Class	**Dataset Class**
First	1
Second	2
Third	3

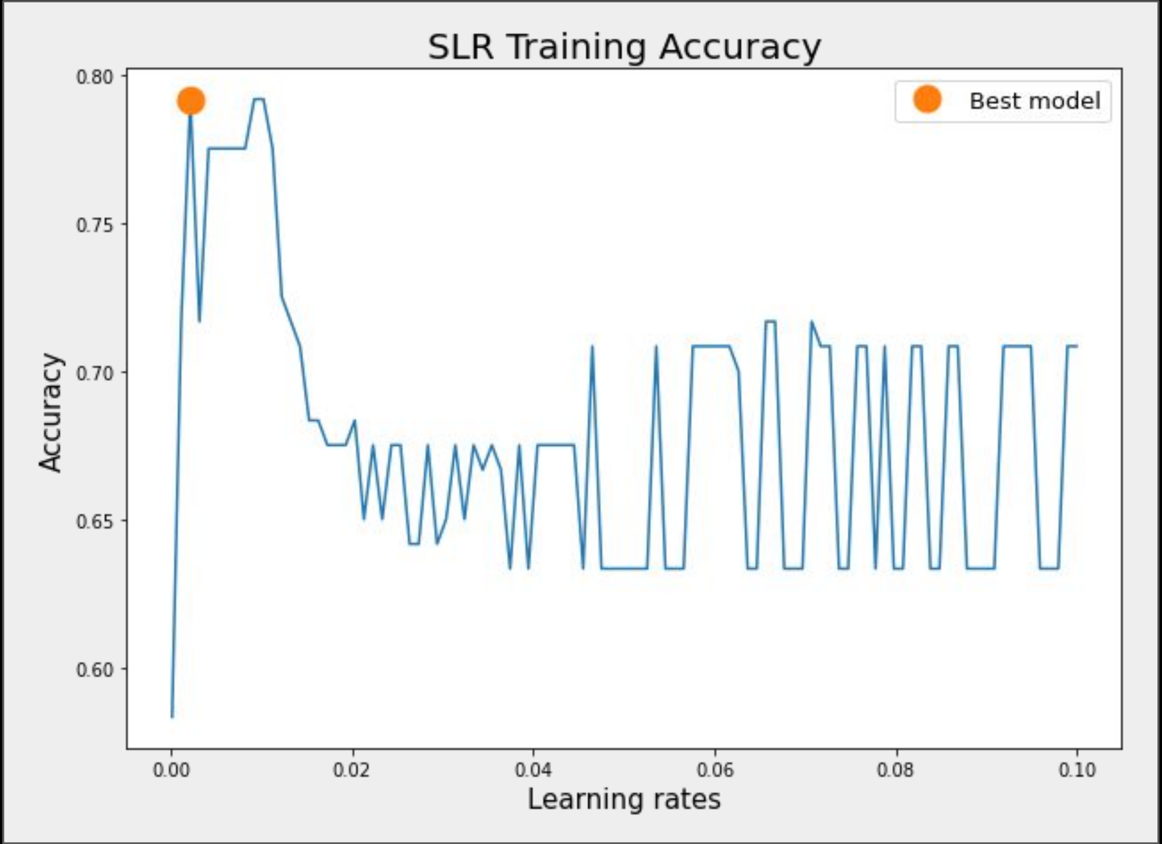
SLR MODEL

- 30 different epochs
- 1K to 30K
- Selected value: 17K
- Learning rate: 0.001

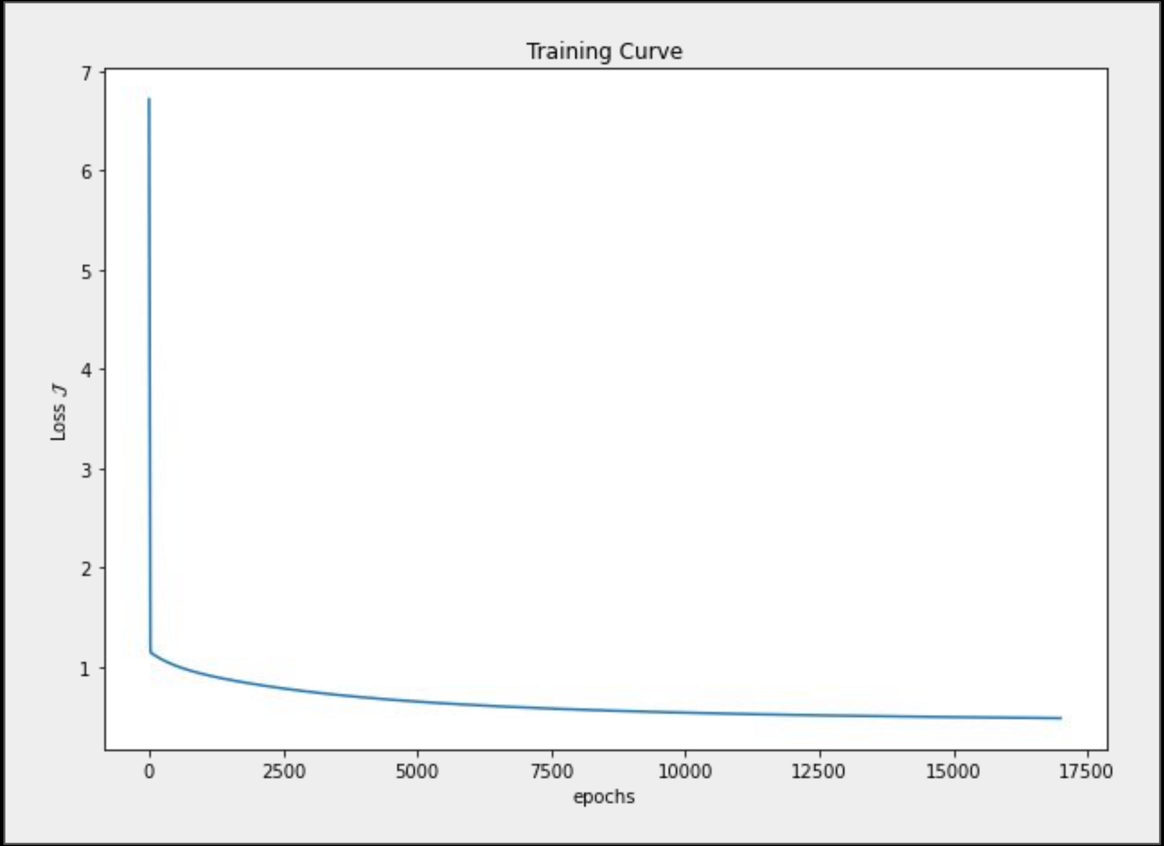


SLR MODEL

- 100 different learning rates
- 0.0001 to 0.1
- Selected value: 0.002118

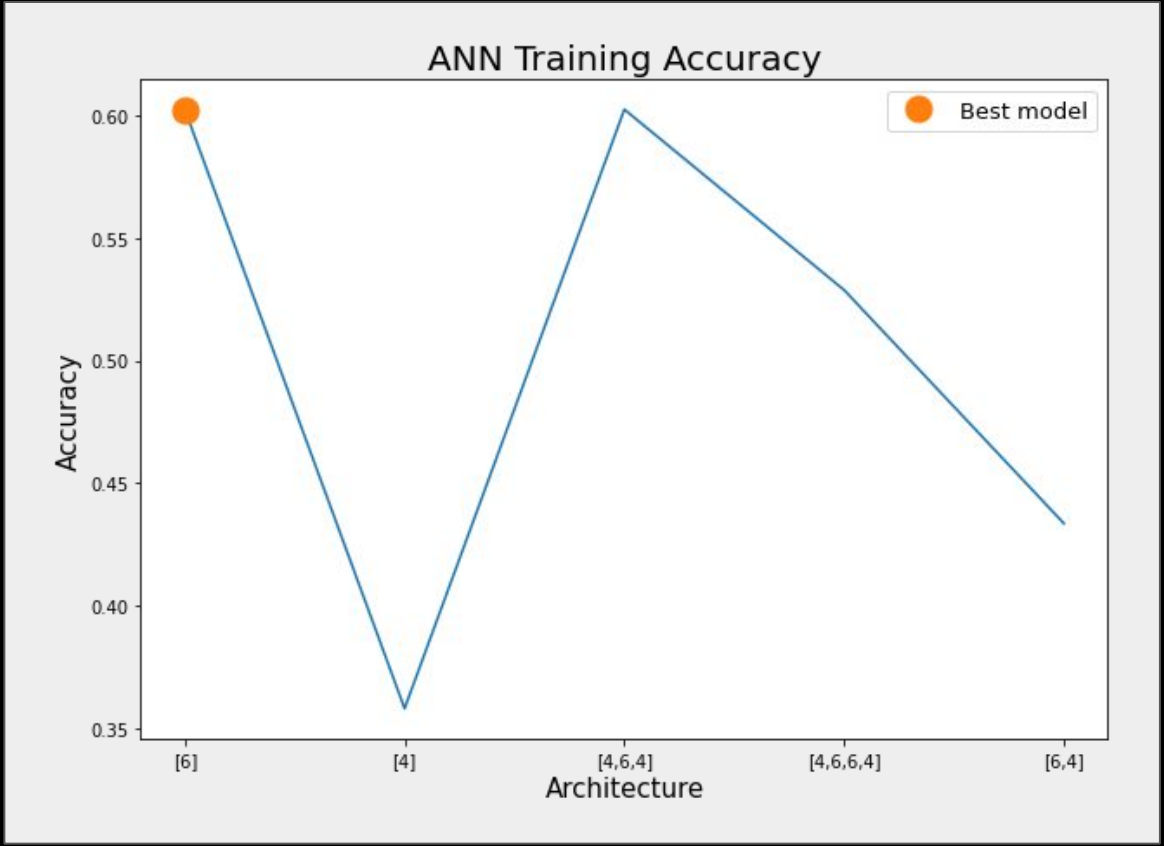


SLR MODEL


Training Curve
Loss $\mathcal{J}$
epochs
0
2500
5000
7500
10000
12500
15000
17500
1
2
3
4
5
6
7

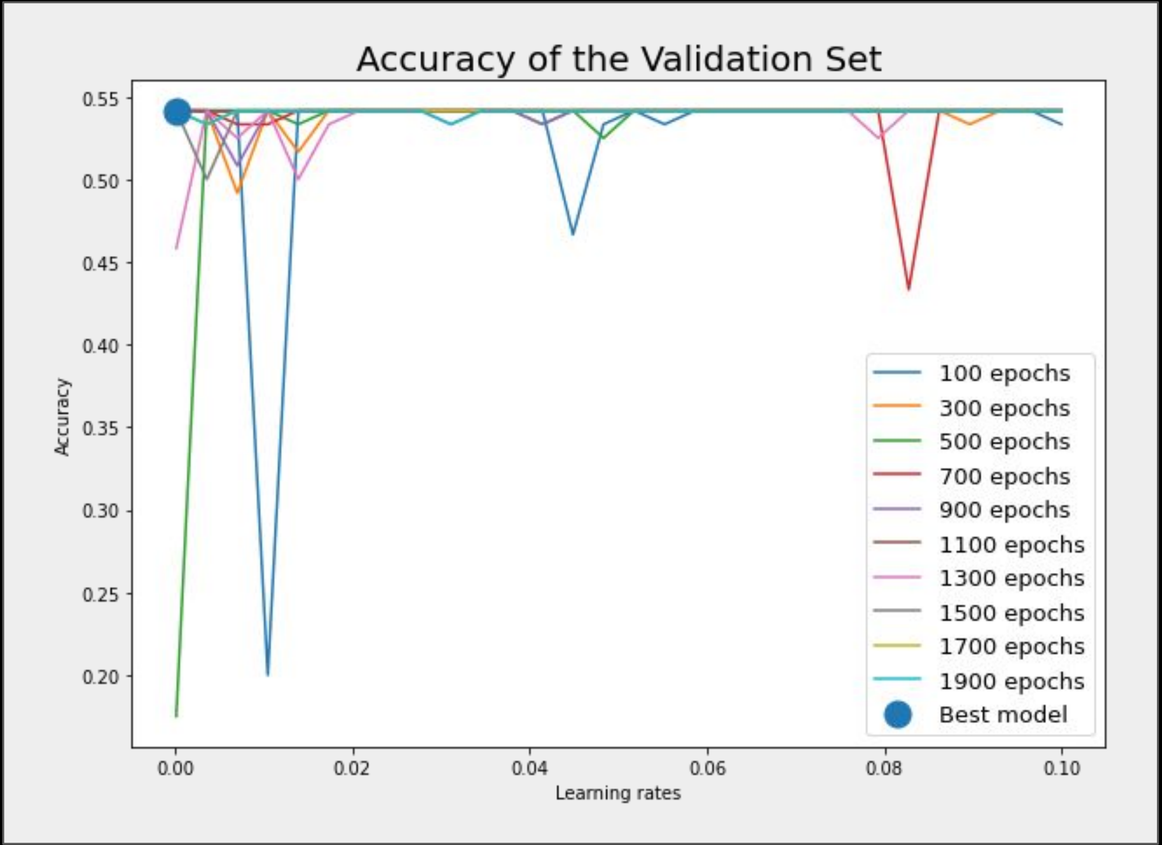
ANN FOR CLASSIFICATION

- 5 different architectures
  - [6]
  - [4]
  - [4,6,4]
  - [4,6,6,4]
  - [6,4]
- Selected model: [6]
- Activation function: sigmoid
- Learning rate: 0.001
- Epochs: 1000

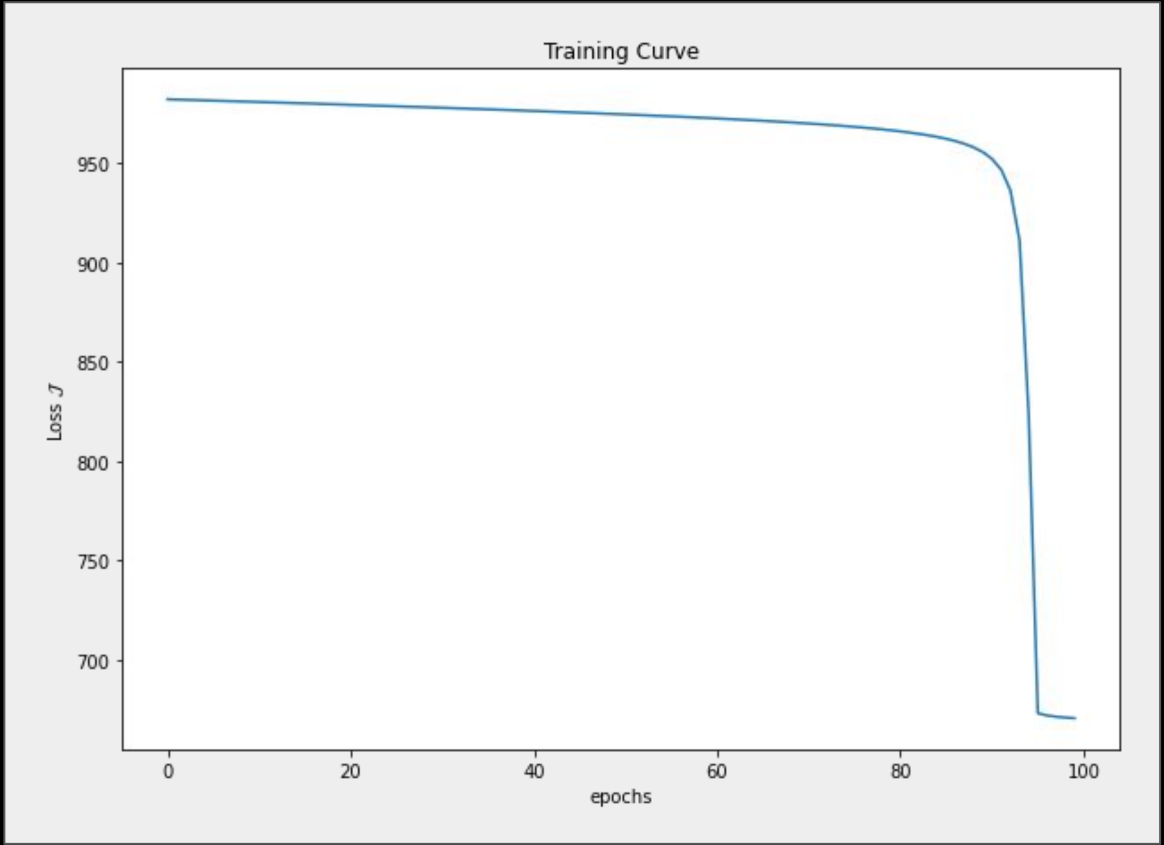


ANN FOR CLASSIFICATION

- Finetune epochs
  - 10 values
  - range(100,2000,200)
  - Selected: 100
- Finetune learning rate
  - 30 values
  - From 0.0001 to 0.1
  - Selected: 0.0001



ANN FOR CLASSIFICATION

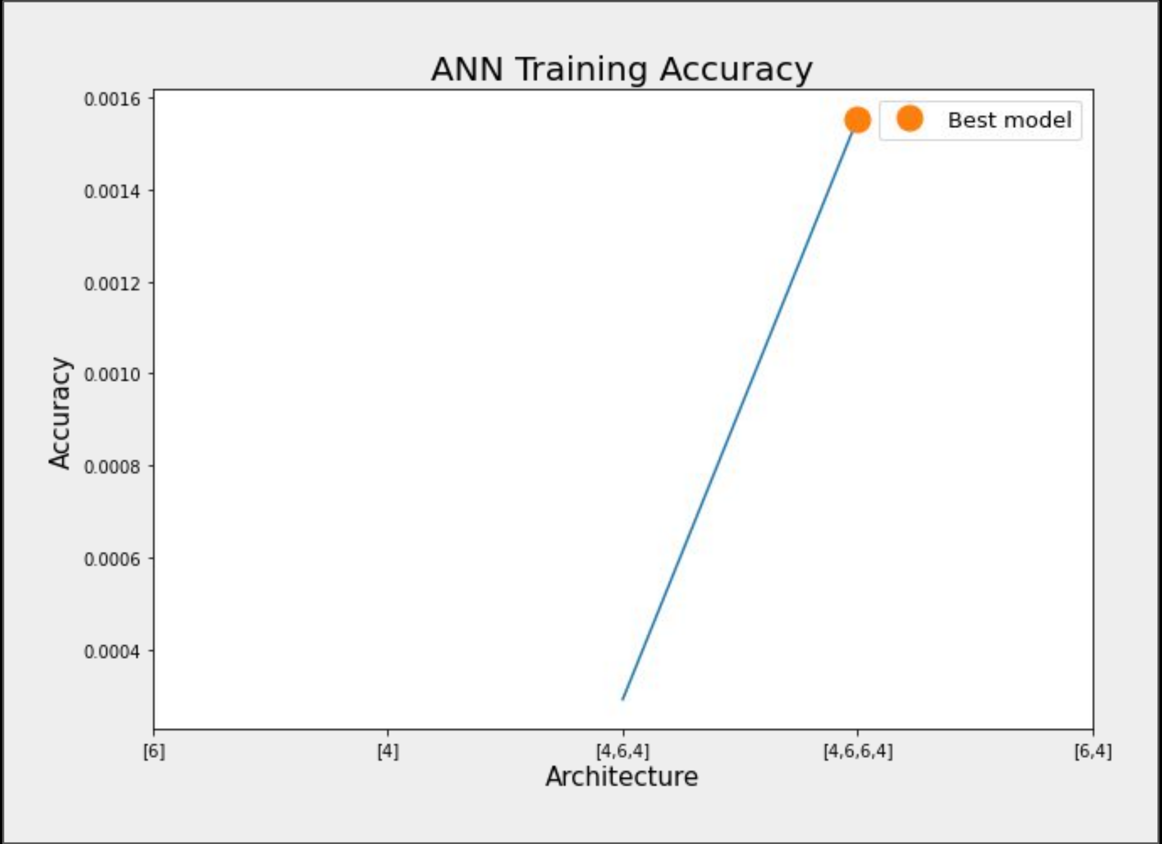


CLASSIFICATION COMPARISON

	Architecture	Epochs	Learning rate	Training accuracy	Validation accuracy	Test accuracy
SLR	-	17,000	0.002118	79.67%	79.16%	76.47%
ANN	[6]	100	0.0001	60.25%	54.16%	57.98%

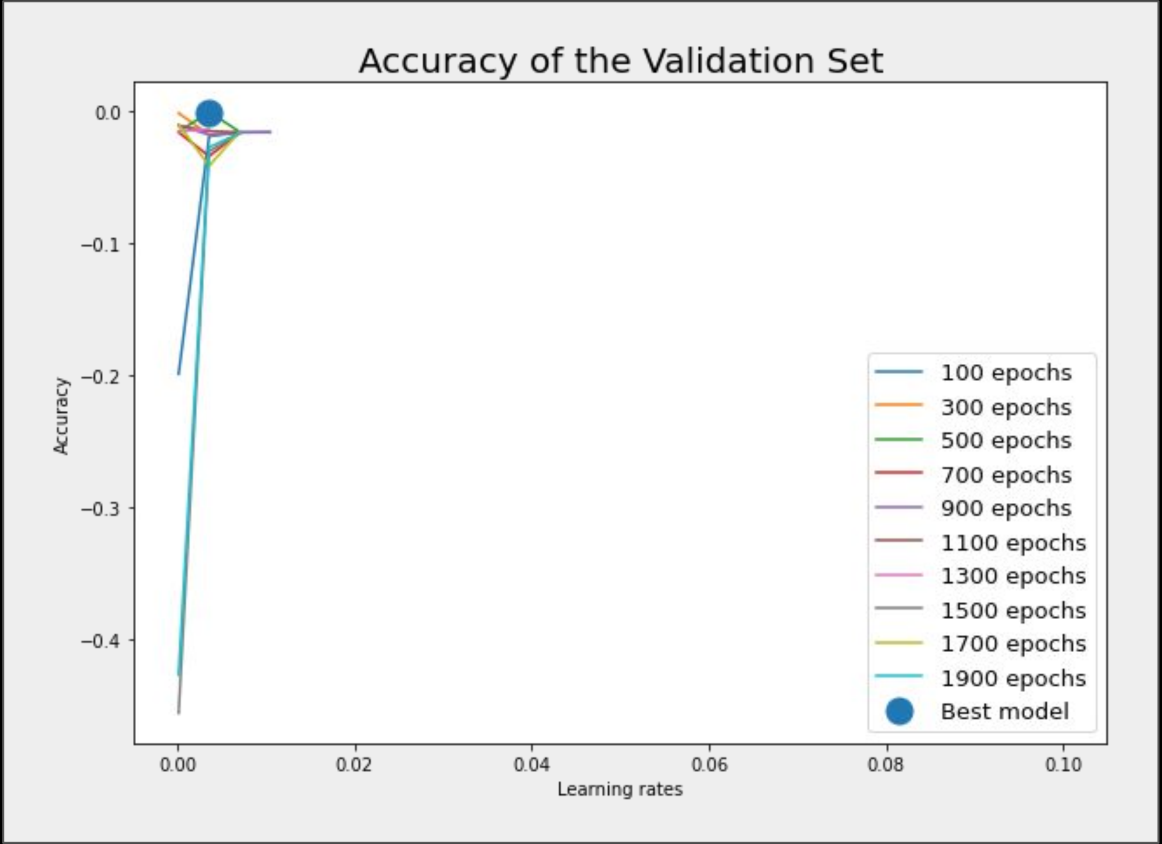
ANN FOR REGRESSION

- 5 different architectures
  - [6]
  - [4]
  - [4,6,4]
  - [4,6,6,4]
  - [6,4]
- Selected model: [4,6,6,4]
- Activation function: [ReLU,tanh,tanh,ReLU]
- Learning rate: 0.001
- Epochs: 1000

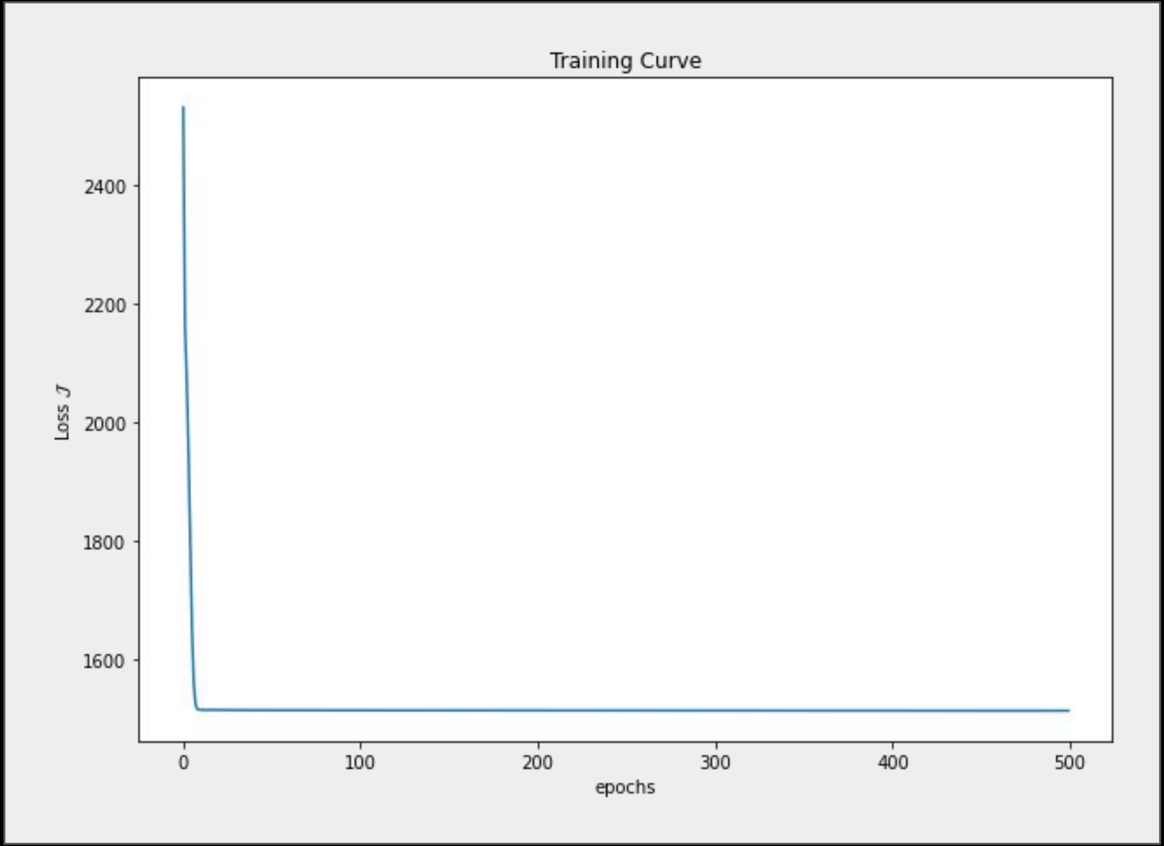


ANN FOR REGRESSION

- Finetune epochs
  - 10 values
  - range(100,2000,200)
  - Selected: 500
- Finetune learning rate
  - 30 values
  - From 0.0001 to 0.1
  - Selected: 0.0035



ANN FOR REGRESSION



REGRESSION RESULTS

	Architecture	Epochs	Learning rate	Training R^2	Validation R^2	Test R^2
Regression ANN	[4,6,6,4]	500	0.0035	0.001554	-0.00082	-0.00386

ANN CLASS MODIFICATION

```python
def one_hot(y,binary):
  if binary:
    return y
  else:
    N = len(y)
    K = len(set(y))
    Y = np.zeros((N,K))
    for i in range(N):
      Y[i, y[i]] = 1

    return Y
```

ANN CLASS MODIFICATION

```python
class ANN():
  def __init__(self, architecture, activations=None, mode=0):
    self.mode = mode
    self.architecture = architecture
    self.activations = activations
    self.L = len(architecture) + 1

  def fit(self, x, y, lr=1e-3, epochs=1e3, show_curve=False, binary=False):
    epochs = int(epochs)
```

ANN CLASS MODIFICATION

```
def fit(self, x, y, lr=1e-3, epochs=1e3, show_curve=False, binary=False):
  epochs = int(epochs)

  #Regression
  if self.mode:
    y = y
  #Classification
  else:
    y = one_hot(y,binary)

  N, D = x.shape
  if binary:
    K = 1
  else:
    K = y.shape[1]
```

ANN CLASS MODIFICATION

```
#Set mode
if self.mode:
  self.a[self.L] = linear
else:
  if binary:
    self.a[self.L] = sigmoid  #Change into sigmoid for binary classification
  else:
    self.a[self.L] = softmax
```

ANN CLASS MODIFICATION

```
for epoch in range(epochs):
  #Forward propagation
  self.__forward__(x)

  if self.mode:
    J[epoch] = OLS(y, self.Z[self.L])
  else:
    if binary:
      J[epoch] = bin_cross_entropy(y, self.Z[self.L])
    else:
      J[epoch] = cross_entropy(y, self.Z[self.L])
```

ANN CLASS MODIFICATION

```
#Backpropagation
dH = (1/N) * (self.Z[self.L]-y)

for l in sorted(self.W.keys(), reverse=True):
  dW = self.Z[l-1].T @ dH
  db = dH.sum(axis=0)

  if binary:
    self.W[l] = self.W[l] - lr * dW
    self.b[l] = self.b[l] - lr * db
  else:
    self.W[l] -= lr * dW
    self.b[l] -= lr * db
```