Predicting Forest Fire Area

Project Presentation STAN47

2020-12-14

Sara Thiringer

The Task

- Lab 4: Pre-processing and comparison between NN and OLS Regression
- Predict the burned area of forest fires using primarily meteorological data
- Data from the northeast region of Portugal
- Described as “difficult” regression problem
- Focus:
  - Follow process performed by Cortez & Morais (2007)
  - Pre processing
  - Evaluation metrics
  - Comparison NN/non-NN and early-NN/modern-NN

Previous Research

“In [Cortez and Morais, 2007], the output 'area' was first transformed with a ln(x+1) function. Then, several Data Mining methods were applied. After fitting the models, the outputs were post-processed with the inverse of the ln(x+1) transform. Four different input setups were used. The experiments were conducted using a 10-fold (cross-validation) x 30 runs. Two regression metrics were measured: MAD and RMSE. “ (Description from UCI ML repo)

“The proposed solution includes only four weather variables (i.e. rain, wind, temperature and humidity) in conjunction with a SVM and it is capable of predicting the burned area of small fires, which constitute the majority of the fire occurrences.” (Cortez & Morais, 2007)

“This study will consider multilayer perceptrons with one hidden layer of H hidden nodes and logistic activation functions and one output node with a linear function.” (Cortez & Morais, 2007)

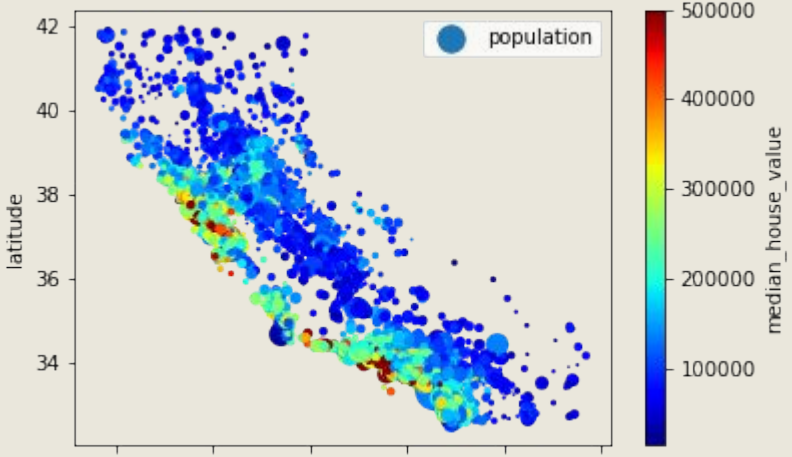
The data

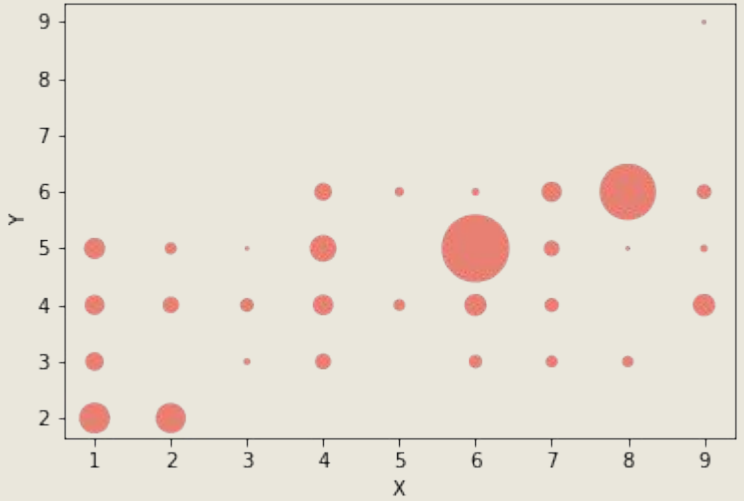
```
data.head()
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0

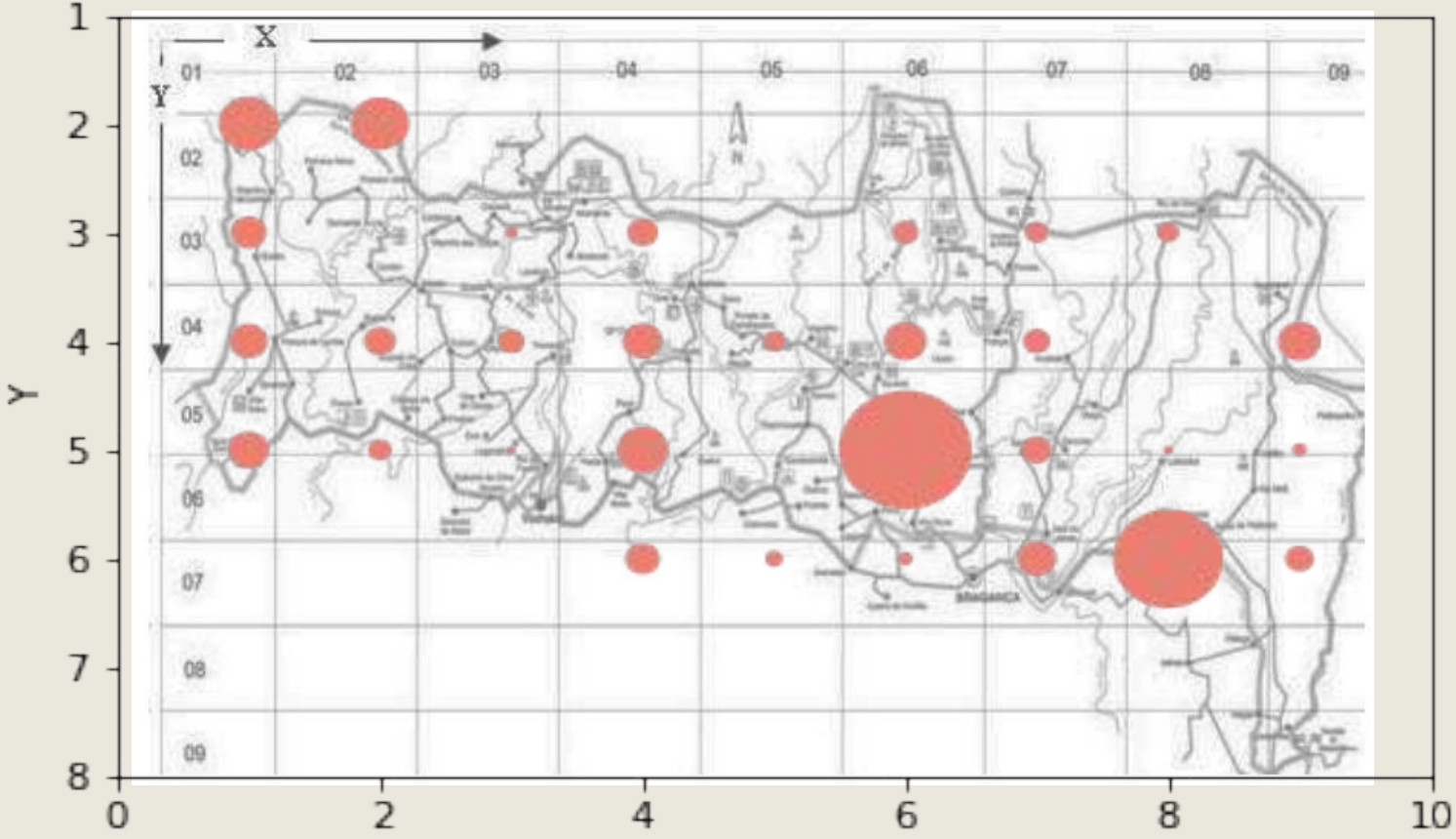
Geographical aspects

```
# Geographical inspection (X and Y is coordinates in the park)
train_fires.plot(kind = 'scatter', x = 'X', y = 'Y', s = 'area')
```





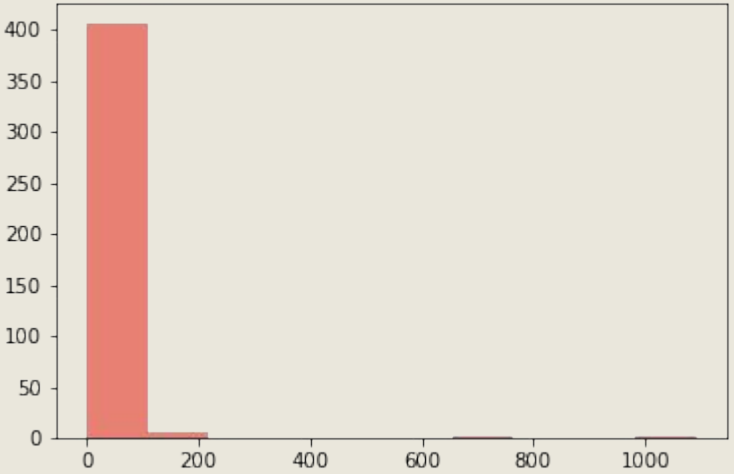
Geographical aspects



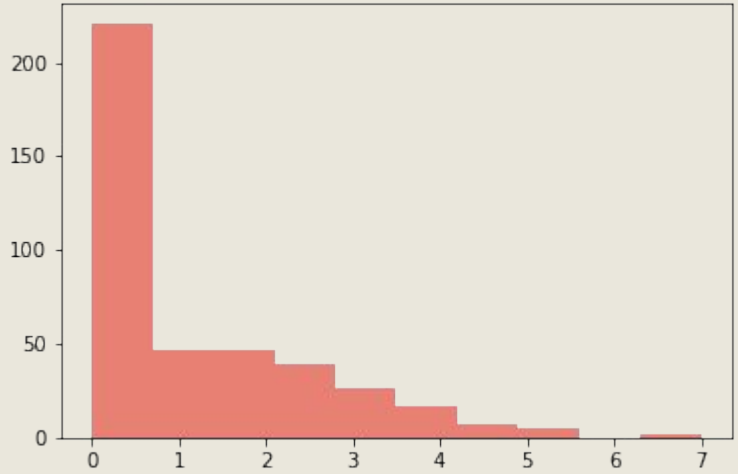
The outcome variable: area (ha)

```
plt.hist(train_fires['area'], color = '#E98074')
plt.savefig('histogram_area', transparent = True)
```

Before transformation



After transformation



Histograms of area before and after ln(x+1) transformation

Categorical variables

- 2 categorical variables
- Month
- Day

```
# Function for converting month to season
def month_to_season(month):
    season = 'other'
    if month in ['dec', 'jan', 'feb']:
        season = 'winter'
    if month in ['mar', 'apr', 'may']:
        season = 'spring'
    if month in ['jun', 'jul', 'aug']:
        season = 'summer'
    if month in ['sep', 'oct', 'nov']:
        season = 'fall'
    return season

fires['season'] = fires['month'].apply(month_to_season)

fires.drop('month', inplace = True, axis = 1)
```

Evaluation metrics

- Trickier for regression than classification if the goal is comparison
- Cortez & Morais (2007) used MAD (MAE) and RMSE
- Custom your own metrics in Keras

$$MAD = 1/N \times \sum_{i=1}^{N} \mid y_i - \hat{y}_i \mid = MAE$$

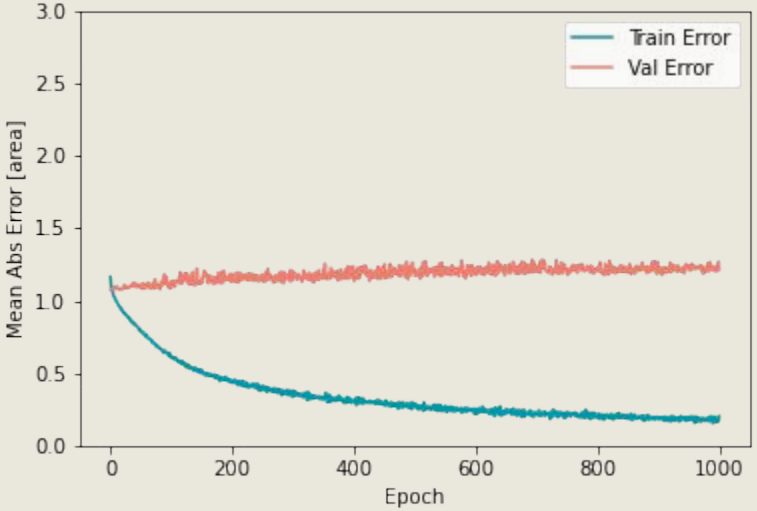
$$RMSE = \sqrt{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 / N}$$

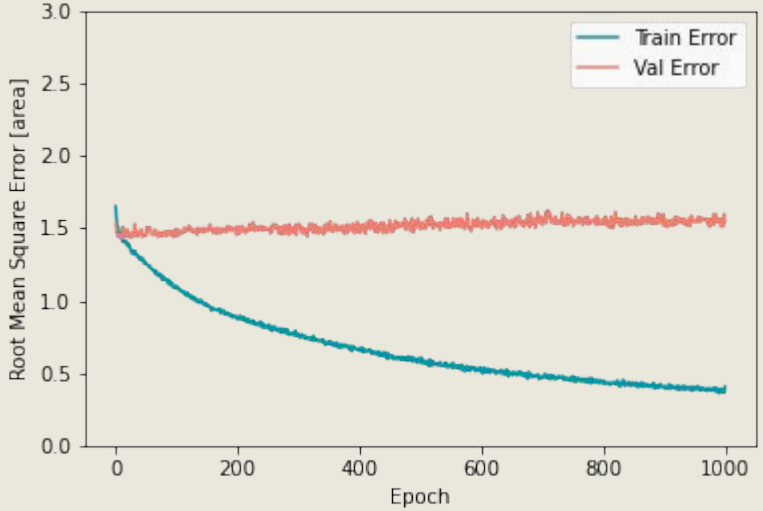
Neural Network Architecture

- 1 hidden layer
- 64 nodes
- ReLU activation

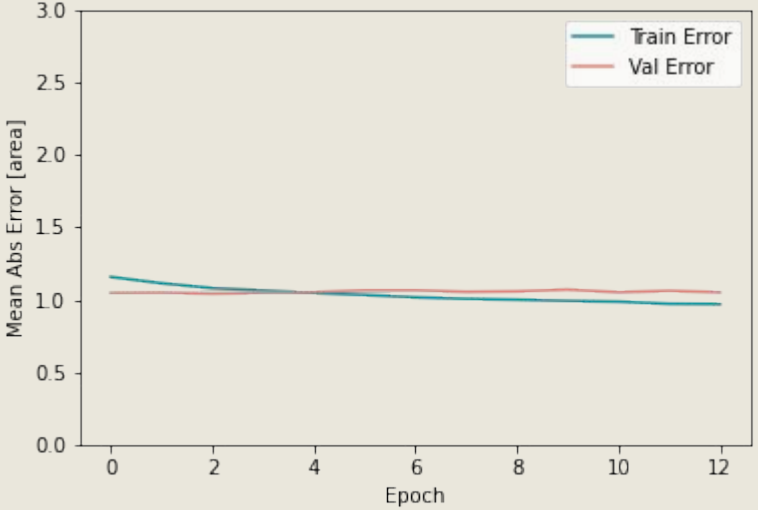
```
model = keras.Sequential([
    layers.Dense(64,
        activation = 'relu',
        input_shape = [len(fires_train_scaled.columns)]),
    layers.Dense(64,
        activation = 'relu'),
    layers.Dense(1)
])

optimizer = tf.keras.optimizers.RMSprop(0.001)

model.compile(loss = 'mae',
    optimizer = optimizer,
    metrics = [tf.keras.metrics.RootMeanSquaredError(),
               'mae'])
```

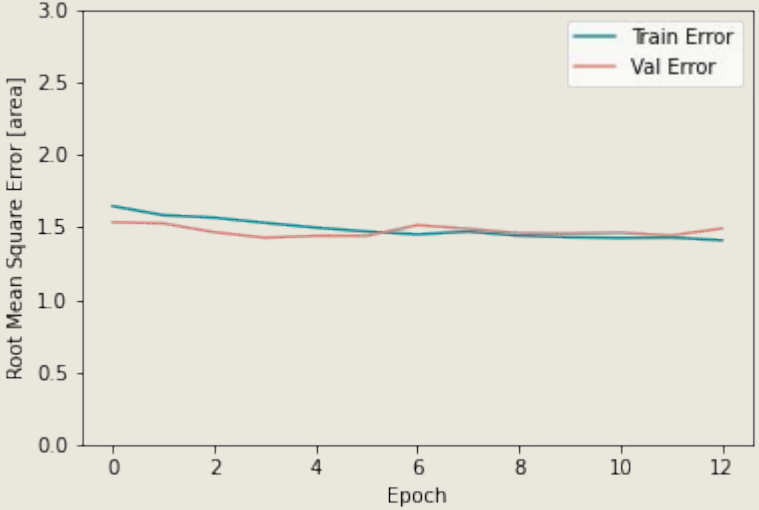
Training Phase





Training Phase with Early Stopping



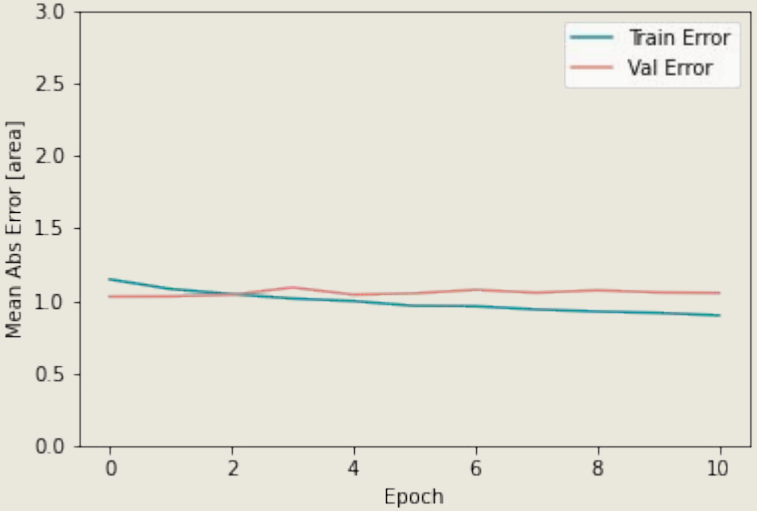


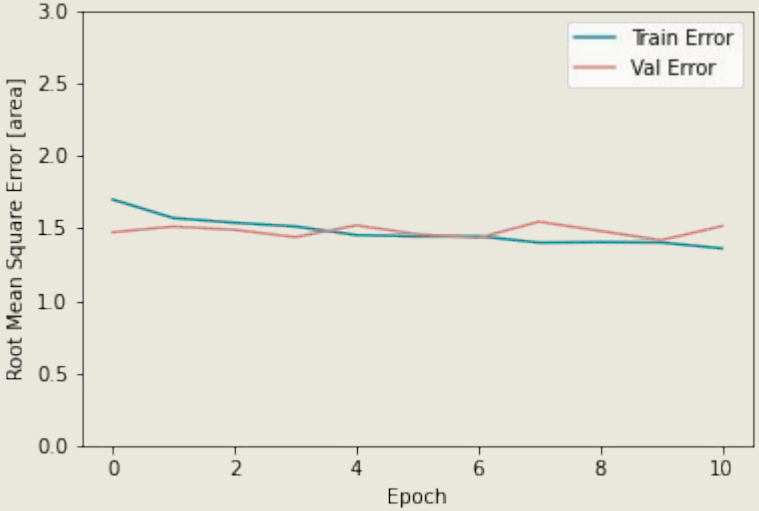
Second Neural Network Architecture

- 3 hidden layers
- 64-100-52-14 nodes
- ReLU activation

```
model2 = keras.Sequential([
    layers.Dense(64,
        activation = 'relu',
        input_shape = [len(fires_train_scaled.columns)]),
    layers.Dense(100,
        activation = 'relu'),
    layers.Dense(52,
        activation = 'relu'),
    layers.Dense(14,
        activation = 'relu'),
    layers.Dense(1)
])
```

Training Phase with Early Stopping





Calculating the Test Results

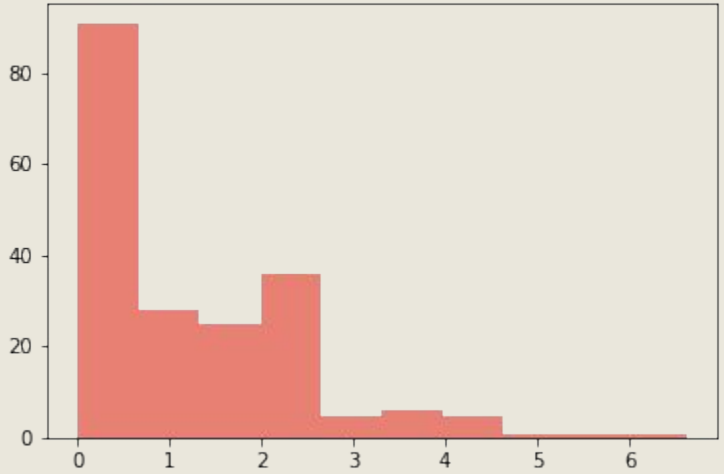
```
# Predict on test data
y_pred = model.predict(fires_test_scaled)

# Transform predictions
y_pred = y_pred.reshape(len(y_pred))
y_pred_final = np.expm1(y_pred)

# Calculate MAE/MAD
np.mean(np.abs(y_test_original - y_pred_final))

# Calculate RMSE
np.sqrt(np.mean((y_test_original - y_pred)**2))
```

Results

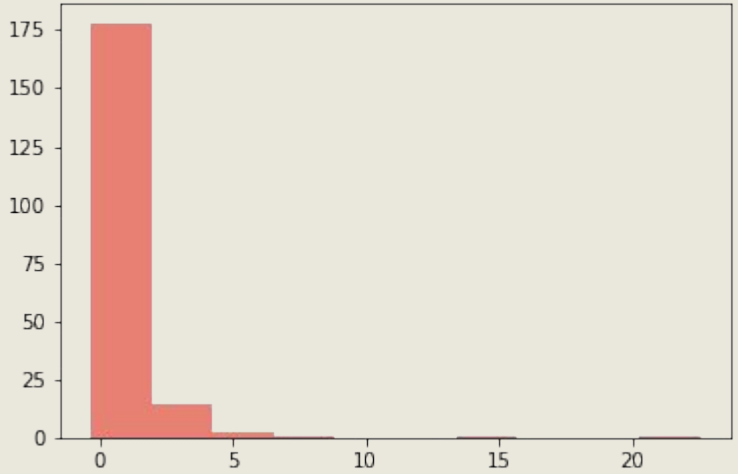
	Model 1	Model 2	Paper
MAE (CV)	13.071	13.106	12.71
MAE (test)	1.194	1.233	N/A
RMSE (CV)	46.923	47.03	63.7
RMSE (test)	1.399	2.298	N/A

Histograms of the true and predicted values

True values



Predicted values



Conclusions and Future Considerations

- Cross-validation is important!
- This task is hard to evaluate due to the nature of the outcome variable
  - Continuous
  - Transformations back and forth
  - Many values of 0
- Trying more architectures
- Taking the log twice (what would this result in?)
- Stratifying the split so that both train and test set includes some of the really high values