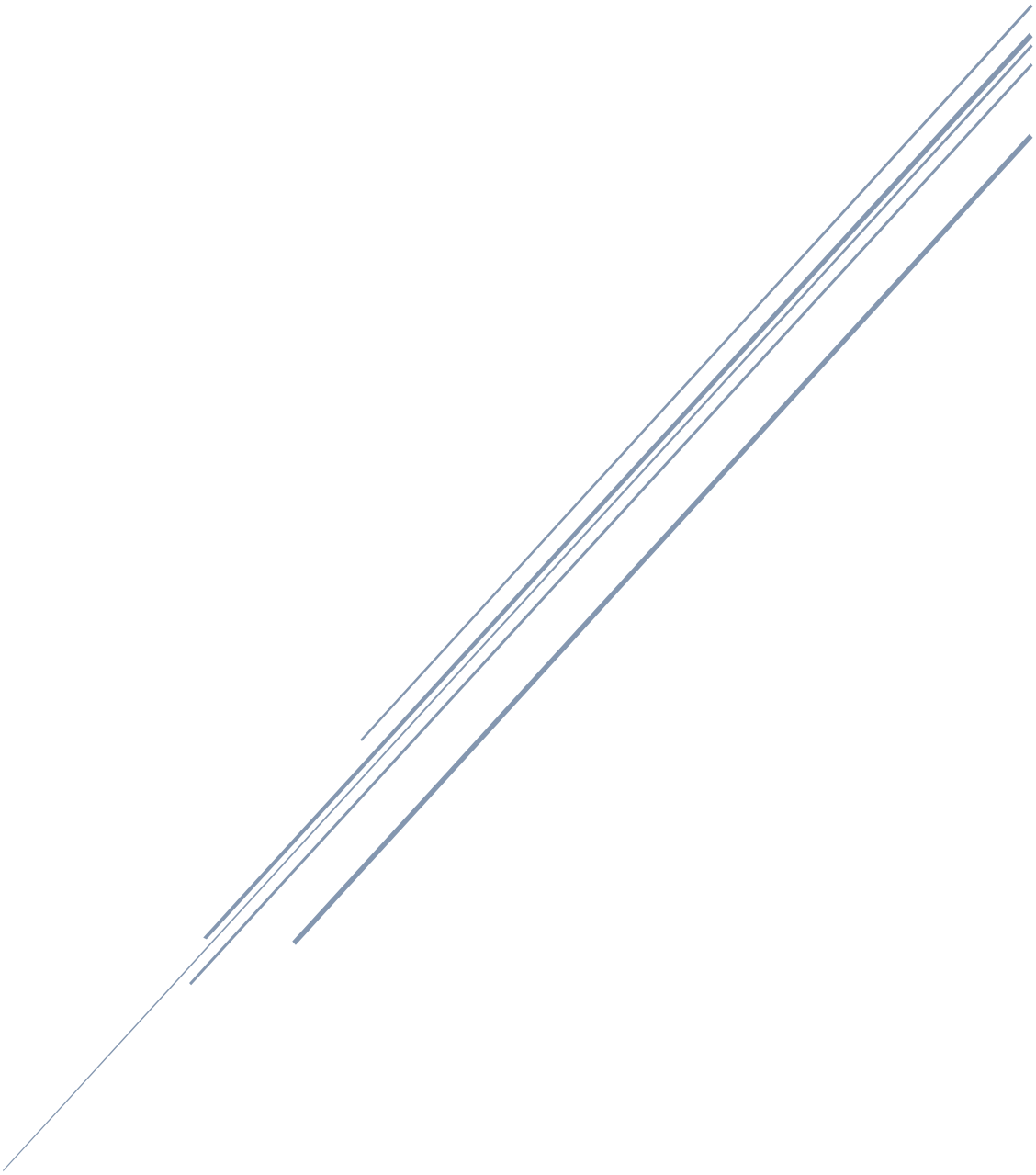
PORTFOLIO

Kianoosh Keshavarzian

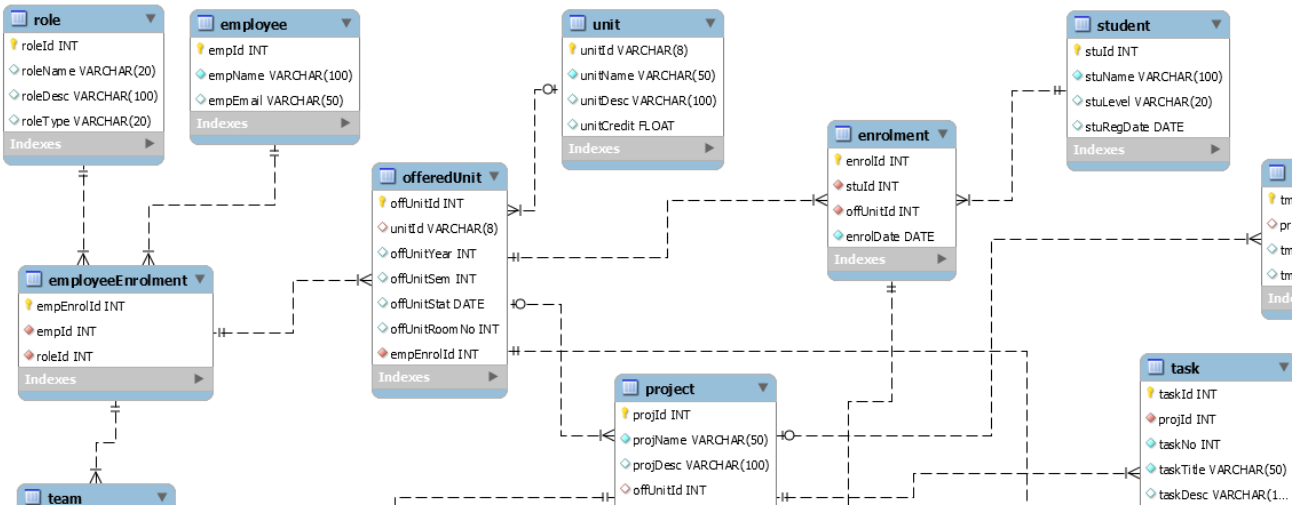


Data Scientist / Analyst

Completed Projects Samples, technologies used and the outcomes

Completed Projects Samples

Database Design and Implementation



Client:

Swinburne University of Technology

Description:

Successfully delivered an advanced, three tier database project for tutorial segment of Swinburne University of Technology based on MySql database and ASP.NET (VB) frontend which was published on AZURE. The project was performed by a group of five and was chosen as the best one out of the three.

Technologies used:

MySql, ASP.NET (VB), AZURE, HTML, CSS, JavaScript

Teamwork:

Group of five

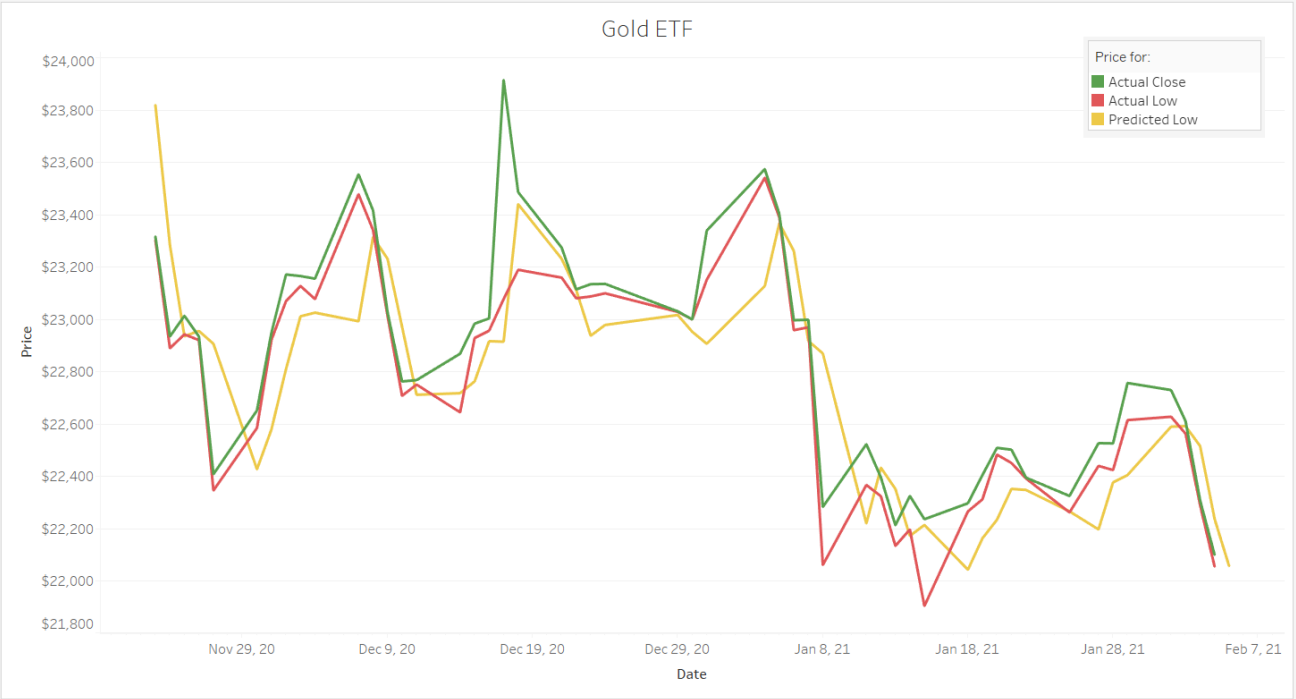
Documents delivered:

Entity Relationship Diagram (ERD), Systems Requirement Specification (SRS), Use Cases, Test Cases, User Documentation, Business Rules & Traceability Matrix

Duration:

Six Month

Stock Market Price Prediction using LSTM



Client:

Bonafide Advice

Description:

Developed Python code to use API to receive businesses statistics and fundamental analysis from Alpha Vantage website, predict the future of the market by implementing a hybrid LSTM model to include variety of markets and handover better predictions on the price movement. Then, populate the Oracle database with analytic results and make webpages connected to the database to deliver the results to the customers.

Technologies used:

API, Python, tensorflow, sklearn, pandasql, Oracle

Teamwork:

Individual

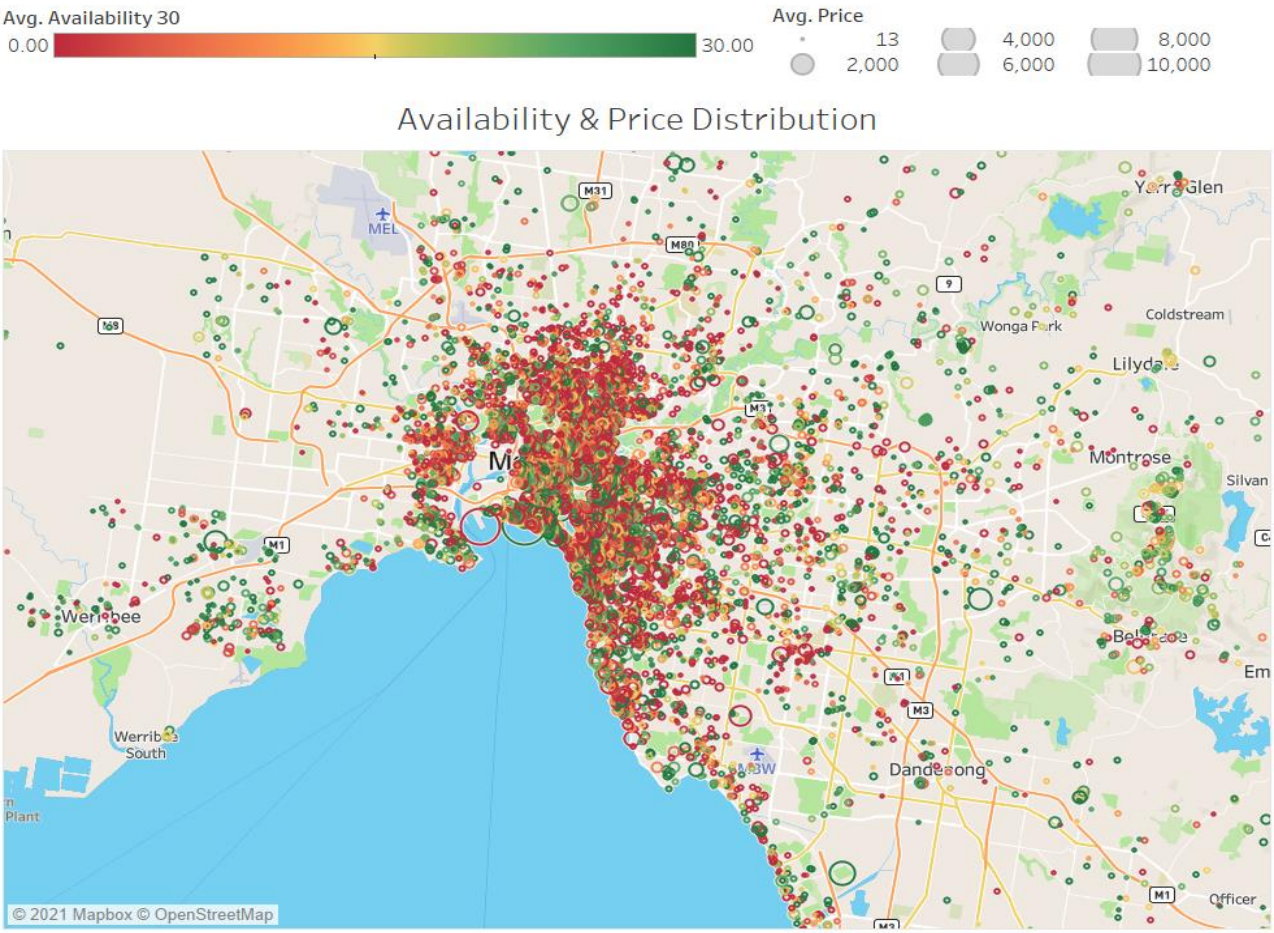
Documents delivered:

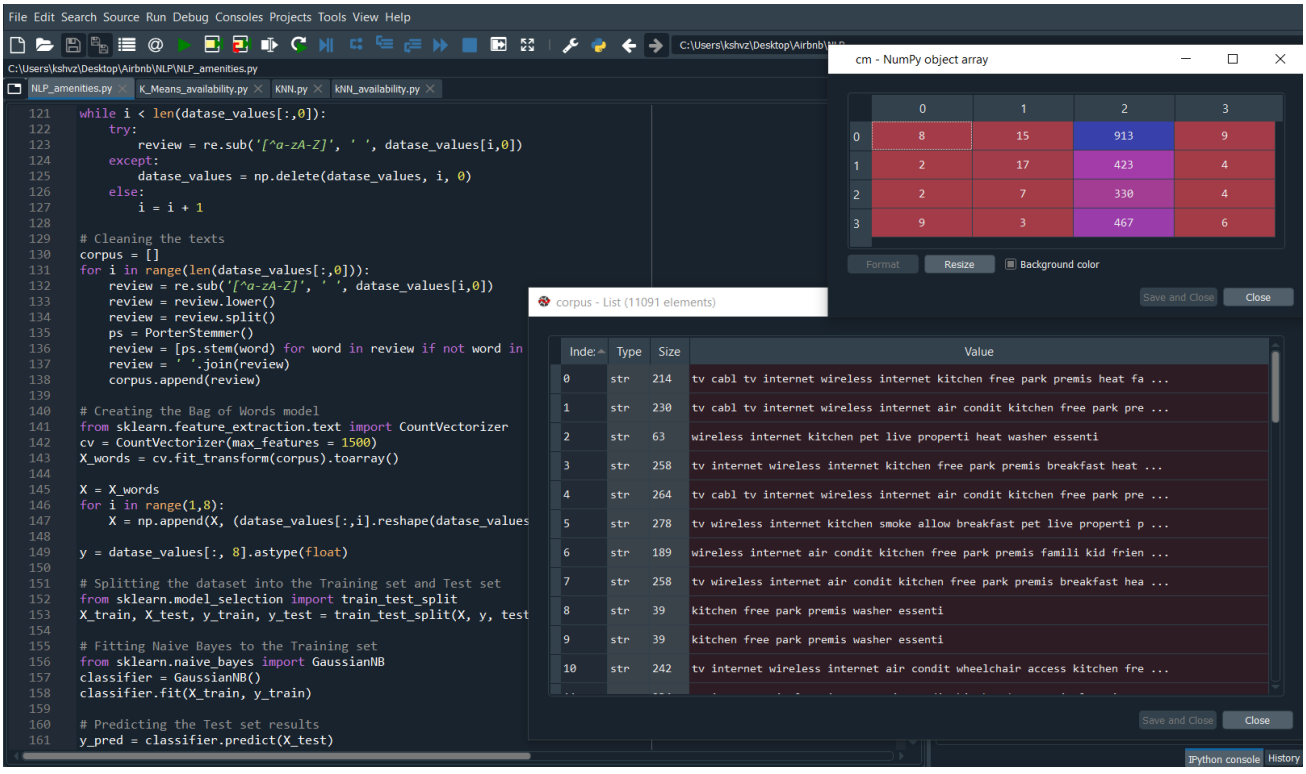
User Manual, Test Cases

Duration:

Four Month

Availability Prediction of Airbnb properties using NLP





Client:

Upwork Client

Description:

Developed an NLP model in Python to predict the number of days that a property is likely to be available base on the amenities and many other categorical valued such as *'does host has profile picture?', 'is location exact?', 'is it instantly bookable?'* etc. The model is using Gaussian Naive Bays and was able to achieve accuracy of over 60%. Other classification models such as decision tree, kernel SVM, KNN, logistic regression, and random forest was used to check the reliability of the results.

Technologies used:

Python, sklearn, nltk, re, Corpus, Porter Stemmer

Teamwork:

Individual

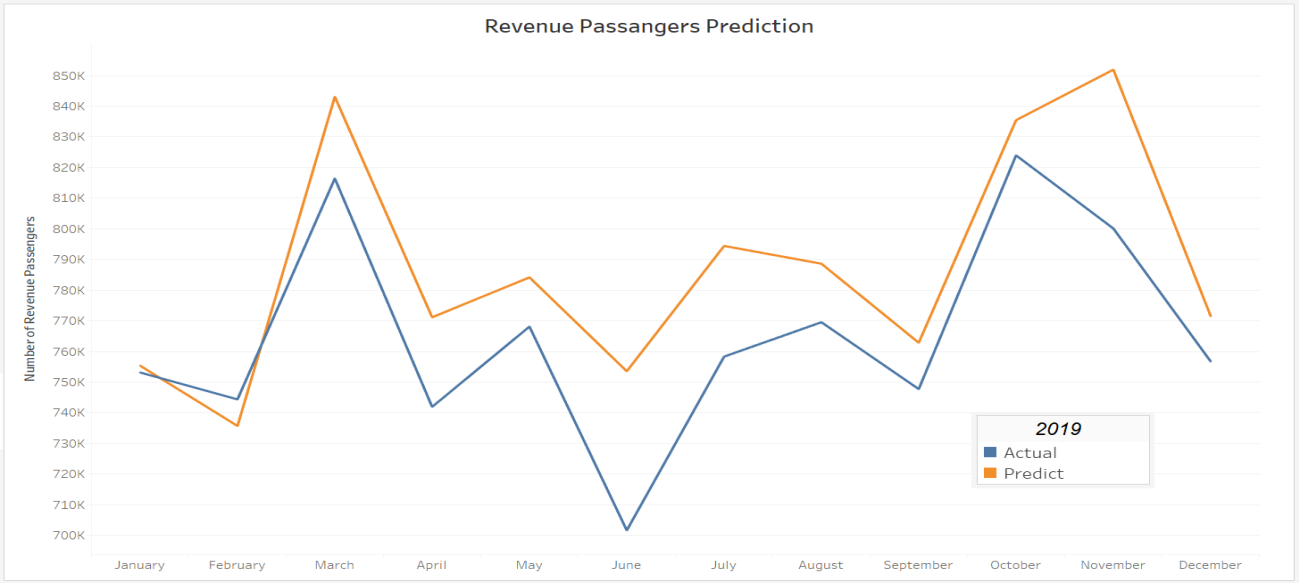
Documents delivered:

ReadMe file, Model description

Duration:

Seven Weeks

Number of Revenue Passenger Prediction using LSTM



Client:

Jetstar Airways

Description:

Predicting the number of revenue passengers for the upcoming month using Multivariate LSTM. The variety of the training set was very limited as the dataset was only contained of the revenue passengers' number for all the cities involved. The number of missing data was also considerably high.

Technologies used:

Python, Tensorflow-Keras, sklearn, pandasql, Tableau

Teamwork:

Group of two

Documents delivered:

Model Description, Tableau Dashboard

Duration:

Two Month

Candlestick Chart conversion to Dataset

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\kshvz

D:\LSTM\stockAnalysis\imageToData\imageToData_GitHub.py

kNN_availability.py | imageToData_GitHub.py*

```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 10 13:34:34 2021

@author: Kianoosh Keshavarzian
"""

from PIL import Image
import datetime
import matplotlib.image as img
import numpy as np
import pandas as pd
from collections import Counter

class imagePrep:
    def __init__(self, fName):
        self.fileName = fName

    #============================================================
    #Converts image to bmp. Needs 'PIL'
    def convbmp(self):
        print('Working on image file: ' + str(self.fileName))
        img_open = Image.open(str(self.fileName))
        img_open.save("currentImage.bmp")
        img_open.close()

    #============================================================
    #Whites out unused parts of the image
    def whiteout(self, xUpperLeft=6, yUpperLeft=6, xLowerRight=1
        img_open = Image.open("currentImage.bmp")
        pixelMap = img_open.load()
        for i in range(int(xUpperLeft), int(xLowerRight)):
            for j in range(int(yUpperLeft), int(yLowerRight)):
                pixelMap[i,j] = (255, 255, 255)
        img_open.save("currentImage.bmp")
        img_open.close()

    #============================================================
    #Reads numbers on vertical axes
    def readVaxes(self, xUpperLeft=1300, yUpperLeft=1, xLowerRig
        img_open = Image.open("currentImage.bmp")
```

D:\LSTM\stockAnalysis\imageToData\imageToData_GitHub.py

kNN_availability.py | imageToData_GitHub.py*

```
        dateTab = pd.DataFrame(dateTab)
        tabOut = pd.DataFrame(tabOut)
        tabOut = tabOut.assign(Date = dateTab[0])
        ohlcVal = pd.DataFrame(ohlcVal)
        tabOut = tabOut.assign(Open=ohlcVal[0], High=ohlcVal[1], Low=o
        print (tabOut)
        return tabOut

#================================================================
# Exports the final table as a .csv file
def main():
    imgp = imagePrep ("2009(1).png")
    imgp.convbmp()
    imgp.whiteout()

    topbotVal = imgp.readVaxes()
    topVal, botVal = topbotVal[0], topbotVal[1]

    dgz = digitize(img.imread('currentImage.bmp'))
    data = dgz.digitImg()

    lines = dgz.grid()
    vLinesX, hLinesY = lines[0], lines[1]

    trim = dgz.trimPos(data, vLinesX)
    begin, end = trim[0], trim[1]

    newData = dgz.trimDigitImg(data, begin, end)
    new_vLinesX = dgz.newVgrid(vLinesX, begin)
    stickEnds = dgz.stkEnds(newData, new_vLinesX)
    ohlcPosition = dgz.ohlcPos(stickEnds, newData)
    ohlcValues = dgz.ohlcVal(ohlcPosition, hLinesY, topVal, botVal)
    datetab = dgz.dateTable(ohlcPosition, stickEnds, int(2009), int(1)
    tabFinal = dgz.finalTable(datetab, ohlcValues)

    tabFinal.to_csv ('OHLC'+'.csv', header=['Date', 'Open', 'High', 'L

if __name__ == "__main__":
    main()
```

Client:

Upwork, SEST

Description:

Developed a package to convert stock market candlestick chart into dataset. The code is using Optical Caracter Recognition (OCR), PIL and Pytesseract packages to read the axes values from the image. It also applies a manual calibration process to remove the background and unnecessary parts of the image. It is able to retrieve data from the charts with only 0.02% deviation and slash a significant portion of spending on raw data.

Technologies used:

API, Python OCR, PIL, Pytesseract, Data Visualization, Image Processing

Teamwork:

Individual

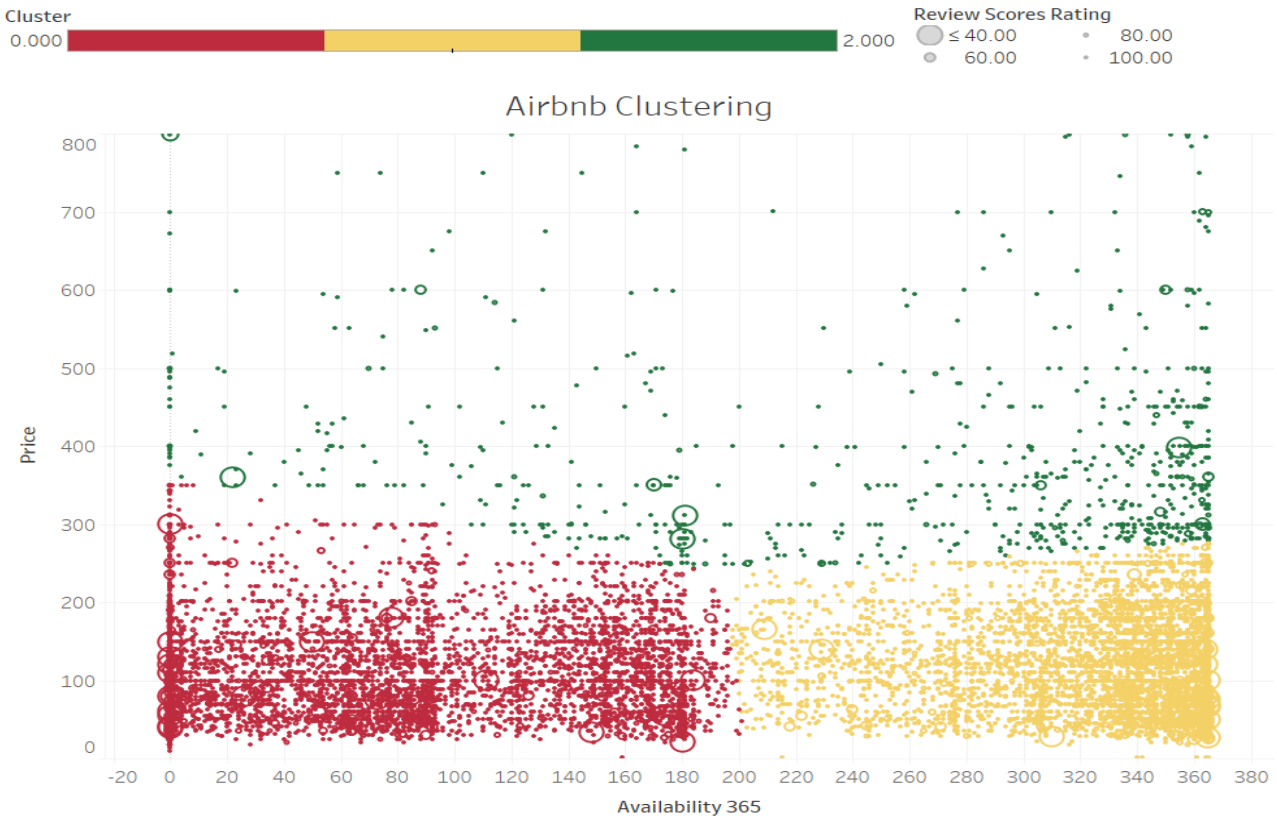
Documents delivered:

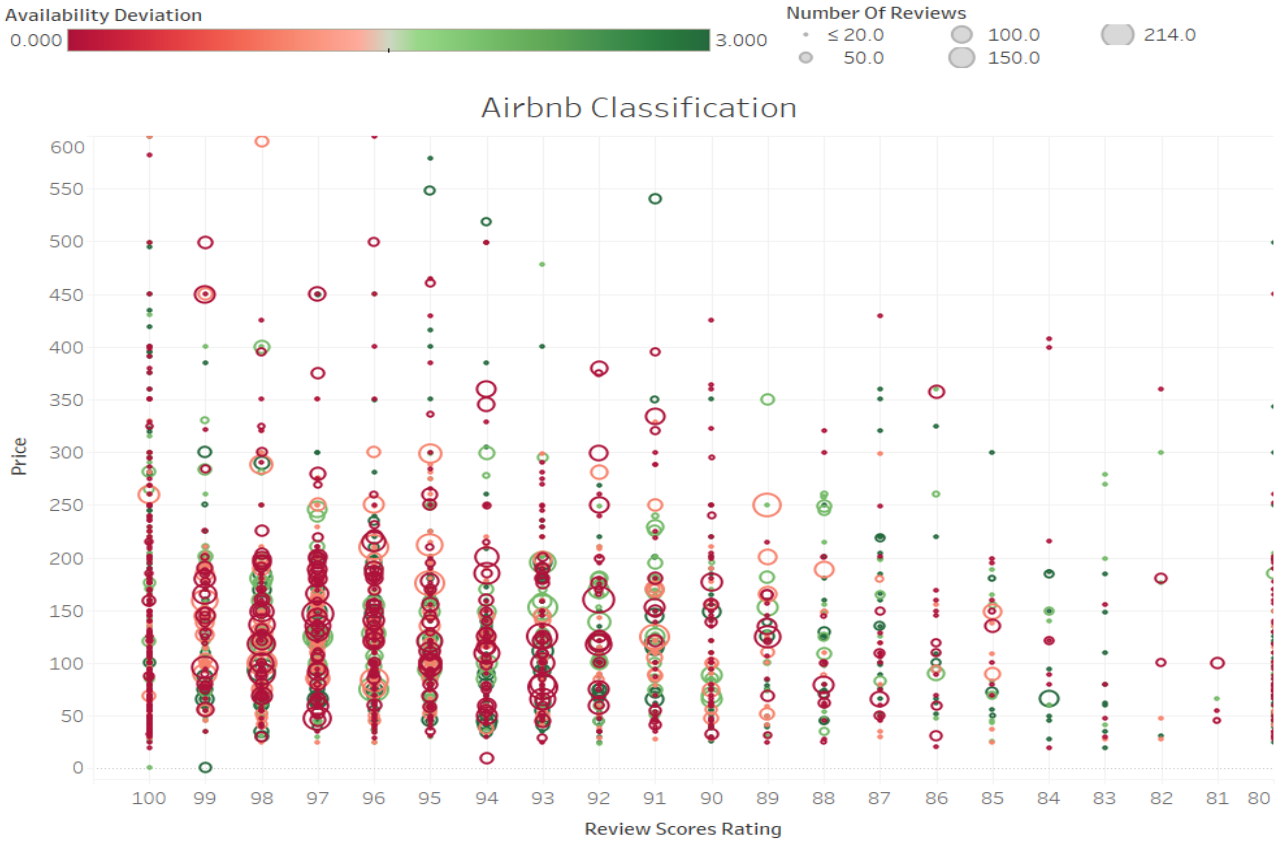
Stock Market Brokers Guide, Image Preparation Guide

Duration:

Three Month

Airbnb Investment Analysis





Client:

Private Investor

Description:

Attained accuracy of 56% in kNN classification model to predict the average number of available days for the property advertised in Airbnb base on *price*, *review score rating* and *number of reviews* whereas the dataset has been extremely dirty. Also categorize the availability against mentioned measures using Python's KMeans clustering. Python code used to cleanse the dataset.

Technologies used:

Excel, Python, Tableau

Teamwork:

Group of two

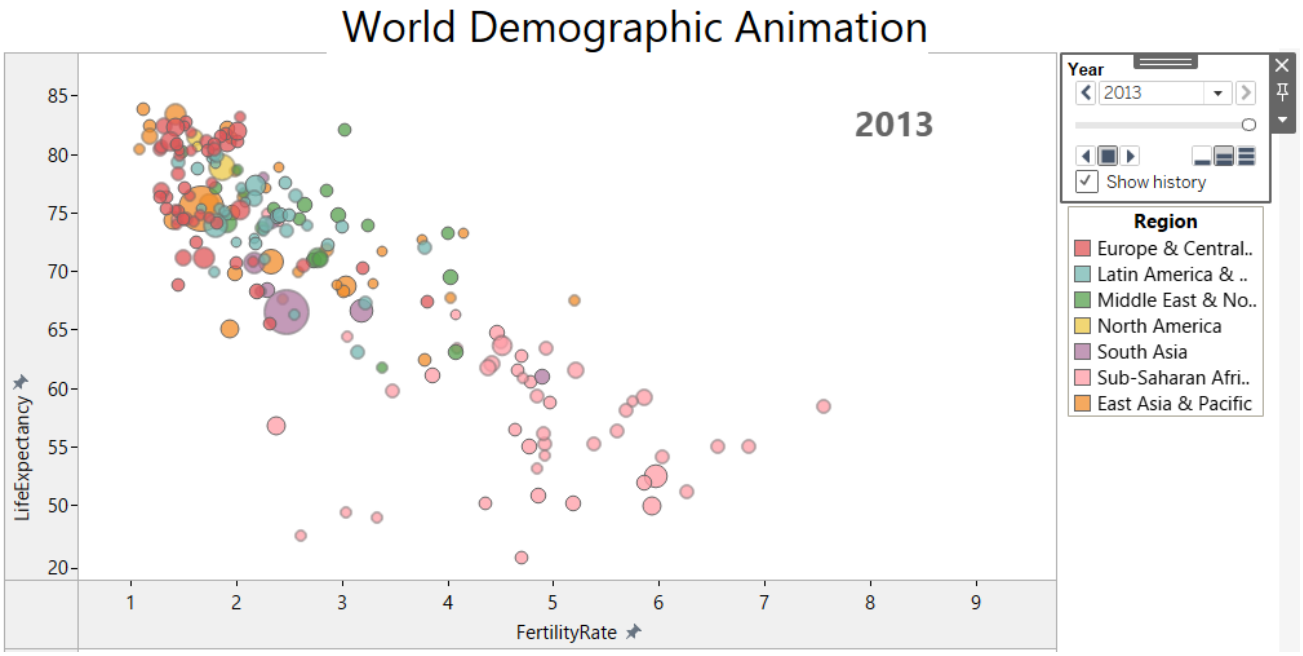
Documents delivered:

Research Report

Duration:

Three Month

Live Data Visualization in Tableau



Client:

Upwork Client

Description:

An advanced dynamic PowerPivot as well as a live animated Tableau visualization on a dataset containing population, fertility rate, life expectancy, country, region and year was performed. The animation moves along the year attribute and visualize the population via size and the region via color as well.

Technologies used:

Excel, PowerPivot, Tableau

Teamwork:

Individual

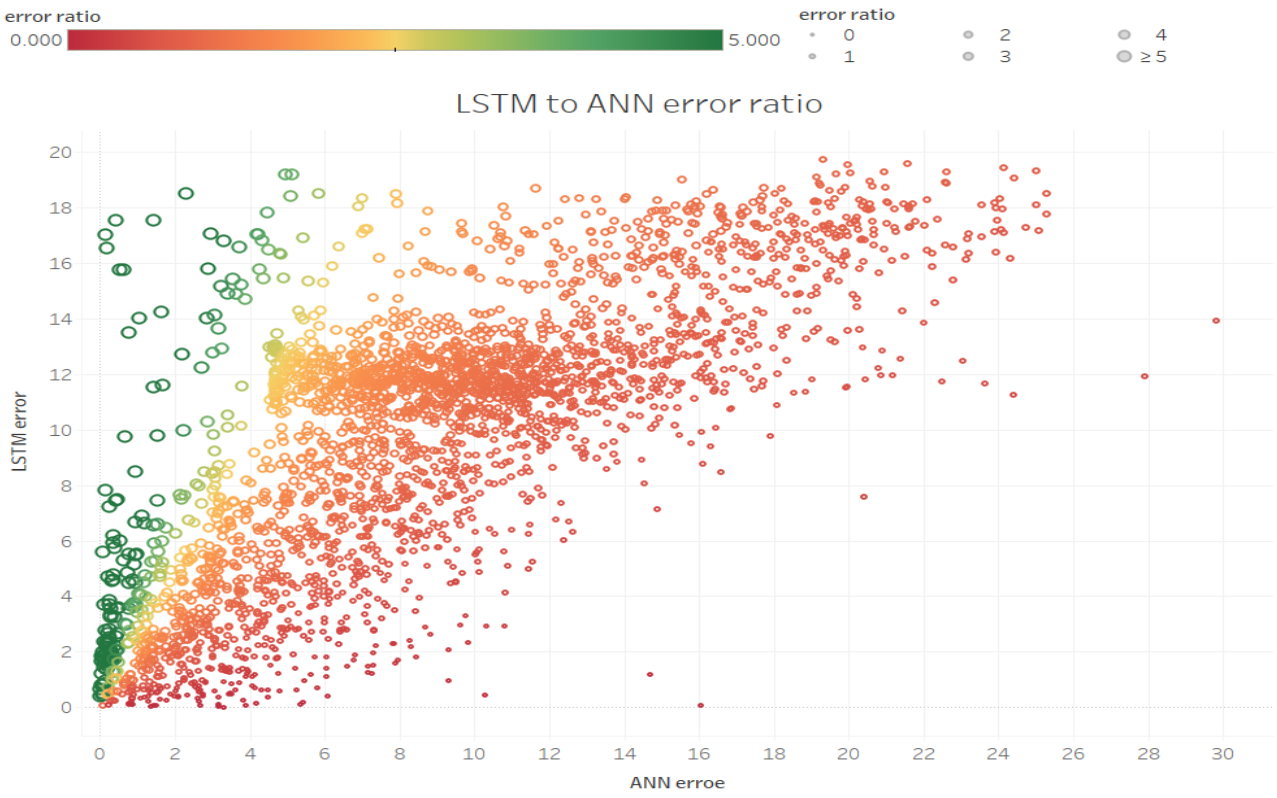
Documents delivered:

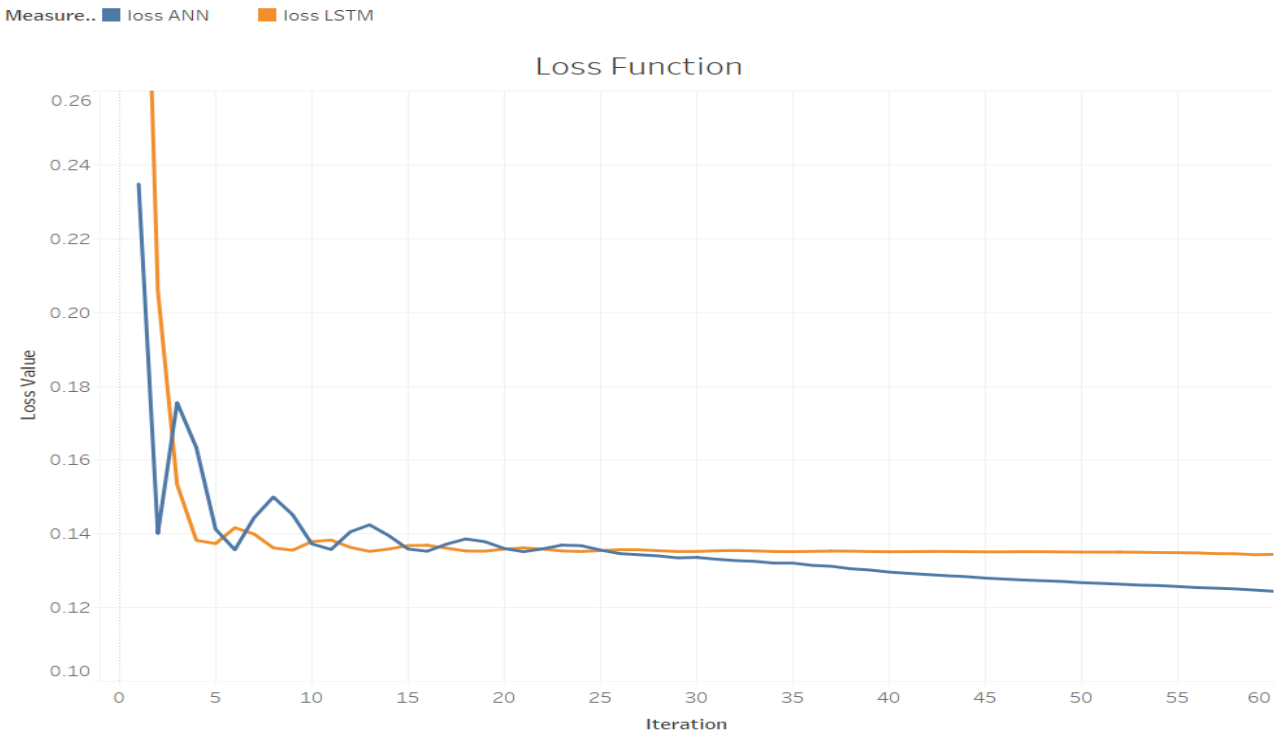
ReadMe file, Animated Video

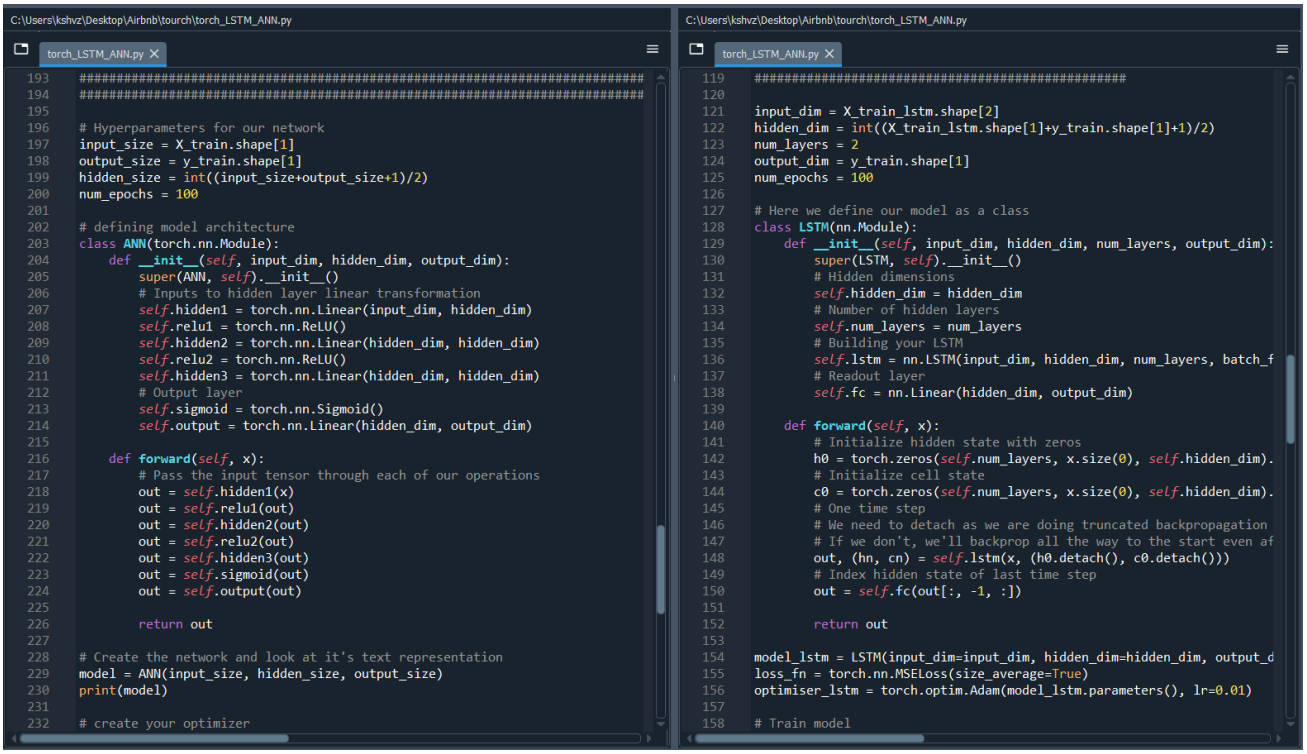
Duration:

One Month

Comparison between Pytorch LSTM and ANN performance on an inconsistent dataset







```
C:\Users\kshvz\Desktop\Airbnb\tourch\torch_LSTM_ANN.py
torch_LSTM_ANN.py
193  ##########################################################################
194  ##########################################################################
195
196  # Hyperparameters for our network
197  input_size = X_train.shape[1]
198  output_size = y_train.shape[1]
199  hidden_size = int((input_size+output_size+1)/2)
200  num_epochs = 100
201
202  # defining model architecture
203  class ANN(torch.nn.Module):
204      def __init__(self, input_dim, hidden_dim, output_dim):
205          super(ANN, self).__init__()
206          # Inputs to hidden layer linear transformation
207          self.hidden1 = torch.nn.Linear(input_dim, hidden_dim)
208          self.relu1 = torch.nn.ReLU()
209          self.hidden2 = torch.nn.Linear(hidden_dim, hidden_dim)
210          self.relu2 = torch.nn.ReLU()
211          self.hidden3 = torch.nn.Linear(hidden_dim, hidden_dim)
212          # Output layer
213          self.sigmoid = torch.nn.Sigmoid()
214          self.output = torch.nn.Linear(hidden_dim, output_dim)
215
216      def forward(self, x):
217          # Pass the input tensor through each of our operations
218          out = self.hidden1(x)
219          out = self.relu1(out)
220          out = self.hidden2(out)
221          out = self.relu2(out)
222          out = self.hidden3(out)
223          out = self.sigmoid(out)
224          out = self.output(out)
225
226          return out
227
228  # Create the network and look at it's text representation
229  model = ANN(input_size, hidden_size, output_size)
230  print(model)
231
232  # create your optimizer
```

```
C:\Users\kshvz\Desktop\Airbnb\tourch\torch_LSTM_ANN.py
torch_LSTM_ANN.py
119  ##########################################################
120
121  input_dim = X_train_lstm.shape[2]
122  hidden_dim = int((X_train_lstm.shape[1]+y_train.shape[1]+1)/2)
123  num_layers = 2
124  output_dim = y_train.shape[1]
125  num_epochs = 100
126
127  # Here we define our model as a class
128  class LSTM(nn.Module):
129      def __init__(self, input_dim, hidden_dim, num_layers, output_dim):
130          super(LSTM, self).__init__()
131          # Hidden dimensions
132          self.hidden_dim = hidden_dim
133          # Number of hidden layers
134          self.num_layers = num_layers
135          # Building your LSTM
136          self.lstm = nn.LSTM(input_dim, hidden_dim, num_layers, batch_f
137          # Readout layer
138          self.fc = nn.Linear(hidden_dim, output_dim)
139
140      def forward(self, x):
141          # Initialize hidden state with zeros
142          h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_dim).
143          # Initialize cell state
144          c0 = torch.zeros(self.num_layers, x.size(0), self.hidden_dim).
145          # One time step
146          # We need to detach as we are doing truncated backpropagation
147          # If we don't, we'll backprop all the way to the start even af
148          out, (hn, cn) = self.lstm(x, (h0.detach(), c0.detach()))
149          # Index hidden state of last time step
150          out = self.fc(out[:, -1, :])
151
152          return out
153
154  model_lstm = LSTM(input_dim=input_dim, hidden_dim=hidden_dim, output_d
155  loss_fn = torch.nn.MSELoss(size_average=True)
156  optimiser_lstm = torch.optim.Adam(model_lstm.parameters(), lr=0.01)
157
158  # Train model
```

Description:

A comparison between LSTM and ANN models' performance on Pytorch was done to decide which model works better on highly inconsistent datasets. The dataset was text heavy and including categorical and numerical values and was required to be cleansed on a massive number of rows. The results showed that ANN converges faster, can achieve smaller loss and prediction error. In addition, LSTM is very intense on CPU and takes significantly more time to train the model.

Technologies used:

Python, Pytorch, Tableau

Teamwork:

Individual

Documents delivered:

Model Description

Duration:

Two weeks