

PORTFOLIO

Kianoosh Keshavarzian



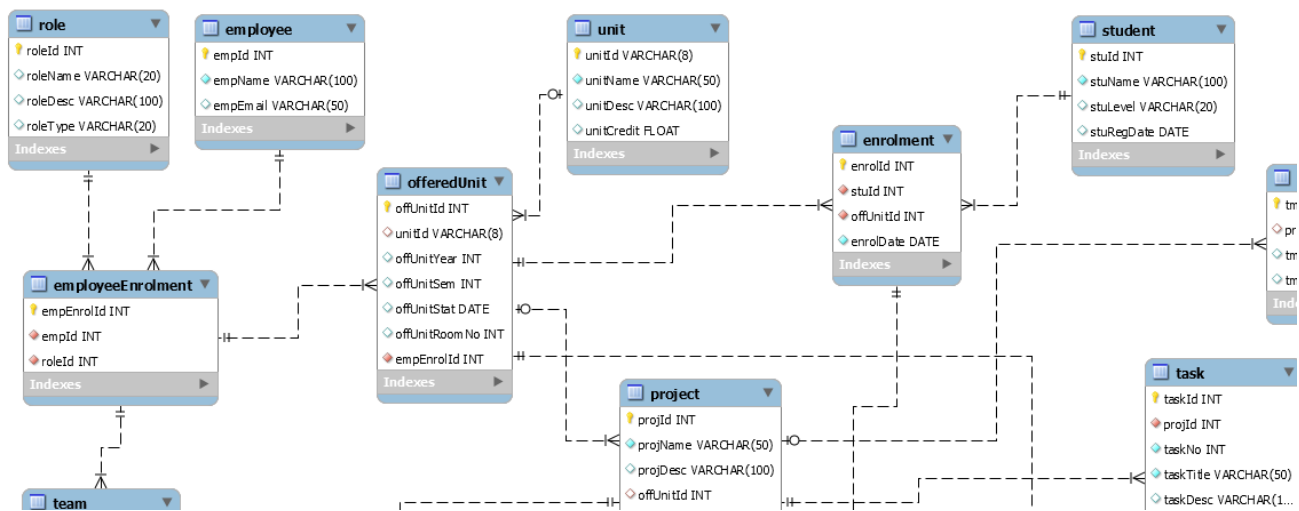
Data Scientist / Analyst

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

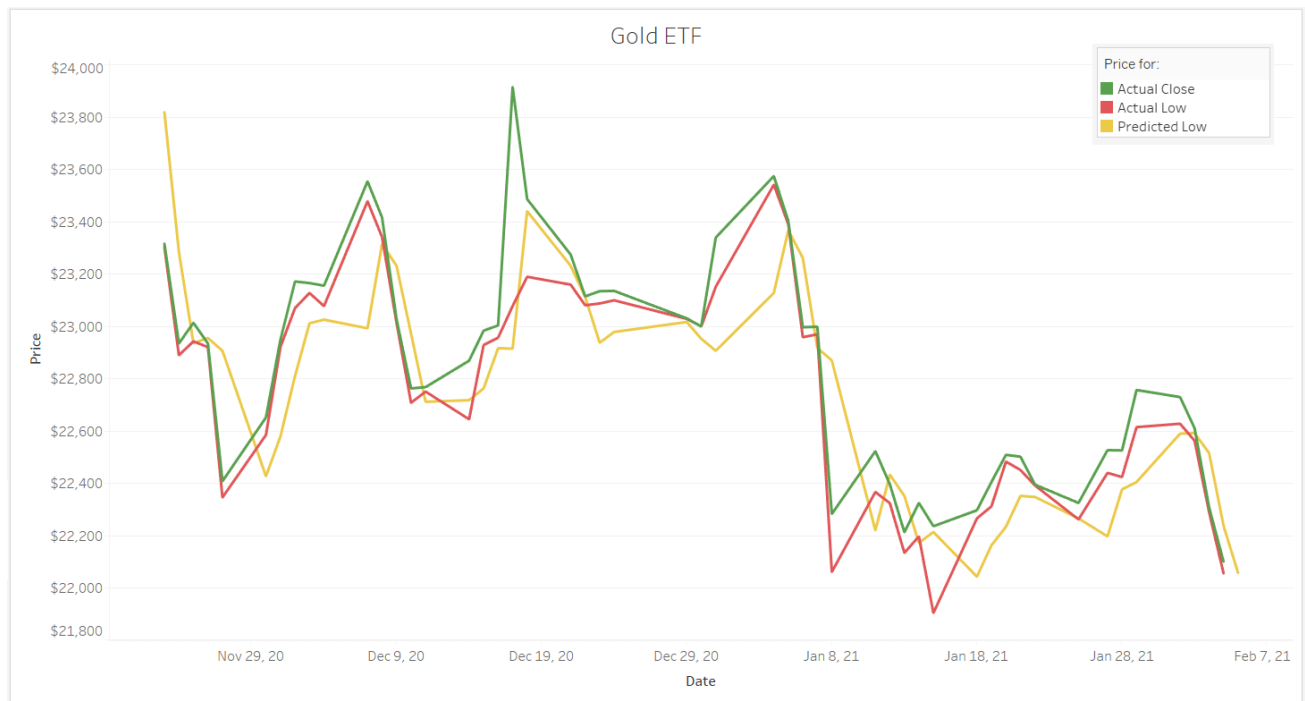
Documents delivered:

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:

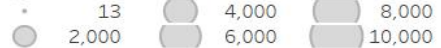
Four Month

Availability Prediction of Airbnb properties using NLP

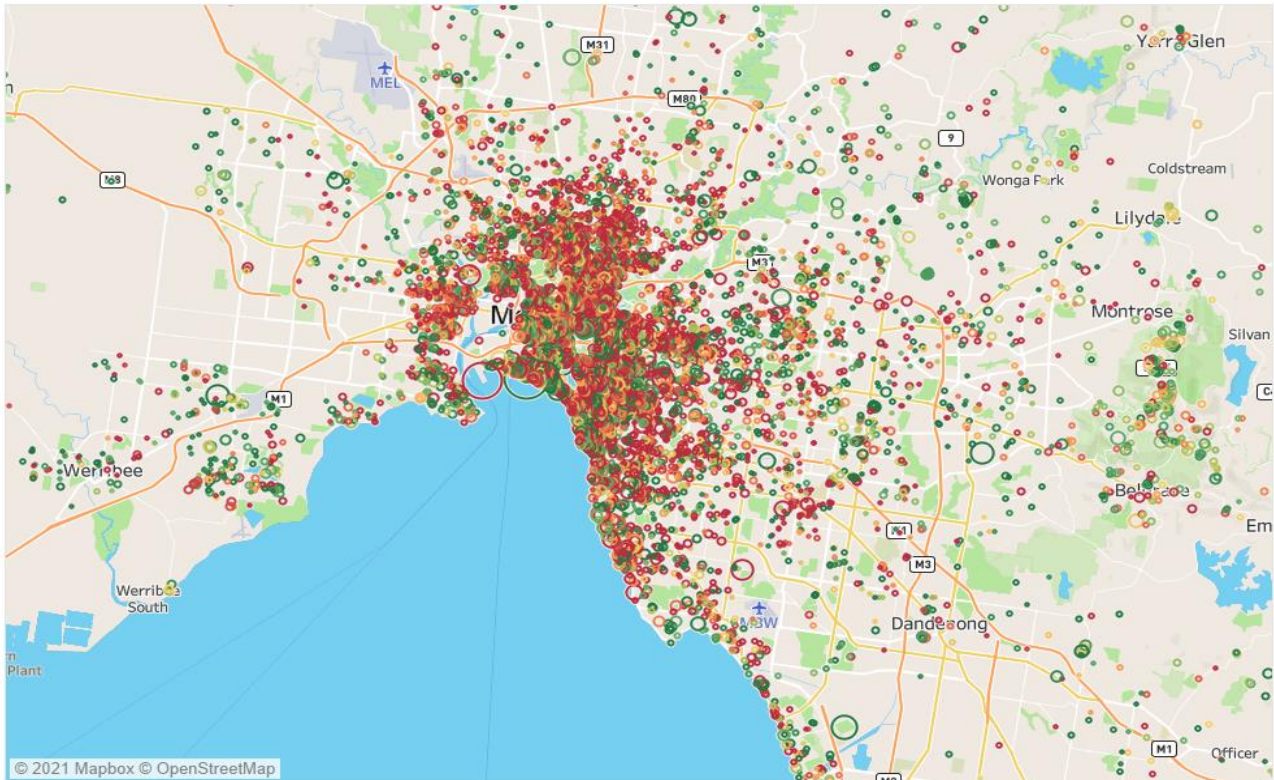
Avg. Availability 30



Avg. Price



Availability & Price Distribution



```
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\kshv\Desktop\Airbnb\NLP\NLP_amenities.py
NLP_amenities.py K-Means_availability.py KNN_availability.py
121 while i < len(data_values[:,0]):
122     try:
123         review = re.sub('[^a-zA-Z]', ' ', data_values[i,0])
124     except:
125         data_values = np.delete(data_values, i, 0)
126     else:
127         i = i + 1
128
129 # Cleaning the texts
130 corpus = []
131 for i in range(len(data_values[:,0])):
132     review = re.sub('[^a-zA-Z]', ' ', data_values[i,0])
133     review = review.lower()
134     review = review.split()
135     ps = PorterStemmer()
136     review = [ps.stem(word) for word in review if not word in stopwords]
137     review = ' '.join(review)
138     corpus.append(review)
139
140 # Creating the Bag of Words model
141 from sklearn.feature_extraction.text import CountVectorizer
142 cv = CountVectorizer(max_features = 1500)
143 X_words = cv.fit_transform(corpus).toarray()
144
145 X = X_words
146 for i in range(1,8):
147     X = np.append(X, (data_values[:,i].reshape(data_values[0,i].size)), axis=1)
148
149 y = data_values[:, 8].astype(float)
150
151 # Splitting the dataset into the Training set and Test set
152 from sklearn.model_selection import train_test_split
153 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
154
155 # Fitting Naive Bayes to the Training set
156 from sklearn.naive_bayes import GaussianNB
157 classifier = GaussianNB()
158 classifier.fit(X_train, y_train)
159
160 # Predicting the Test set results
161 y_pred = classifier.predict(X_test)
```

cm - NumPy object array

	0	1	2	3
0	8	15	913	9
1	2	17	423	4
2	2	7	330	4
3	9	3	467	6

corpus - List (11091 elements)

Index	Type	Size	Value
0	str	214	tv cabl tv internet wireless internet kitchen free park premis heat fa ...
1	str	230	tv cabl tv internet wireless internet air condit kitchen free park pre ...
2	str	63	wireless internet kitchen pet live properti heat washer essenti
3	str	258	tv internet wireless internet kitchen free park premis breakfast heat ...
4	str	264	tv cabl tv internet wireless internet air condit kitchen free park pre ...
5	str	278	tv wireless internet kitchen smoke allow breakfast pet live properti p ...
6	str	189	wireless internet air condit kitchen free park premis famili kid frien ...
7	str	258	tv wireless internet air condit kitchen free park premis breakfast hea ...
8	str	39	kitchen free park premis washer essenti
9	str	39	kitchen free park premis washer essenti
10	str	242	tv internet wireless internet air condit wheelchair access kitchen fre ...

Client:

Freelance 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

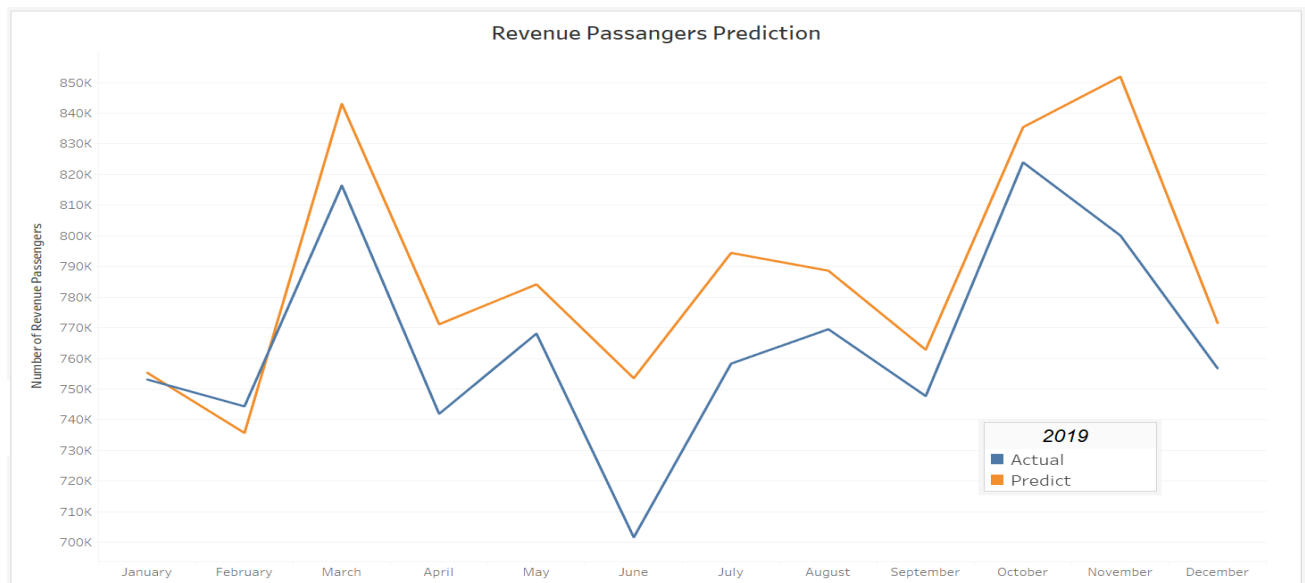
Documents delivered:

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
D:\LSTM\stockAnalysis\imageToData\imageToData_GitHub.py
kNN_availability.py x imageToData_GitHub.py x
1  # -*- coding: utf-8 -*-
2  """
3  Created on Sat Apr 10 13:34:34 2021
4
5  @author: Kianoosh Keshavarzian
6  """
7
8  from PIL import Image
9  import datetime
10 import matplotlib.image as img
11 import numpy as np
12 import pandas as pd
13 from collections import Counter
14
15 class imagePrep:
16     def __init__(self, fName):
17         self.fileName = fName
18
19     #Converts image to bmp. Needs 'PIL'
20     def convbmp(self):
21         print('Working on image file: ' + str(self.fileName))
22         img_open = Image.open(str(self.fileName))
23         img_open.save("currentImage.bmp")
24         img_open.close()
25
26     #Whites out unused parts of the image
27     def whiteout(self, xUpperLeft=6, yUpperLeft=6, xLowerRight=1
28         img_open = Image.open("currentImage.bmp")
29         pixelMap = img_open.load()
30         for i in range(int(xUpperLeft), int(xLowerRight)):
31             for j in range(int(yUpperLeft), int(yLowerRight)):
32                 pixelMap[i,j] = (255, 255, 255)
33         img_open.save("currentImage.bmp")
34         img_open.close()
35
36     #Reads numbers on vertical axes
37     def readVaxes(self, xUpperLeft=1300, yUpperLeft=1, xLowerRig
38         img_open = Image.open("currentImage.bmp")
39
400
401 dateTab = pd.DataFrame(dateTab)
402 tabOut = pd.DataFrame(tabOut)
403 tabOut = tabOut.assign(Date = dateTab[0])
404 ohlcVal = pd.DataFrame(ohlcVal)
405 tabOut = tabOut.assign(Open=ohlcVal[0], High=ohlcVal[1], Low=0
406 print (tabOut)
407 return tabOut
408
409 # Exports the final table as a .csv file
410 def main():
411     imgp = imagePrep ("2009(1).png")
412     imgp.convbmp()
413     imgp.whiteout()
414
415     topbotVal = imgp.readVaxes()
416     topVal, botVal = topbotVal[0], topbotVal[1]
417
418     dgz = digitize(img.imread('currentImage.bmp'))
419     data = dgz.digitImg()
420
421     lines = dgz.grid()
422     vLinesX, hLinesY = lines[0], lines[1]
423
424     trim = dgz.trimPos(data, vLinesX)
425     begin, end = trim[0], trim[1]
426
427     newData = dgz.trimDigitImg(data, begin, end)
428     new_vLinesX = dgz.newVgrid(vLinesX, begin)
429     stickEnds = dgz.stkEnds(newData, new_vLinesX)
430     ohlcPosition = dgz.ohlcPos(stickEnds, newData)
431     ohlcValues = dgz.ohlcVal(ohlcPosition, hLinesY, topVal, botVal)
432     datetab = dgz.dateTable(ohlcPosition, stickEnds, int(2009), int(1)
433     tabFinal = dgz.finalTable(datetab, ohlcValues)
434
435     tabFinal.to_csv ('OHLC'+'.csv', header=['Date', 'Open', 'High', 'L
436
437 if __name__ == "__main__":
438     main()
```


Client:

Freelance Client

Description:

Developed a package to convert stock market candlestick chart into dataset. The code is using Optical Character 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

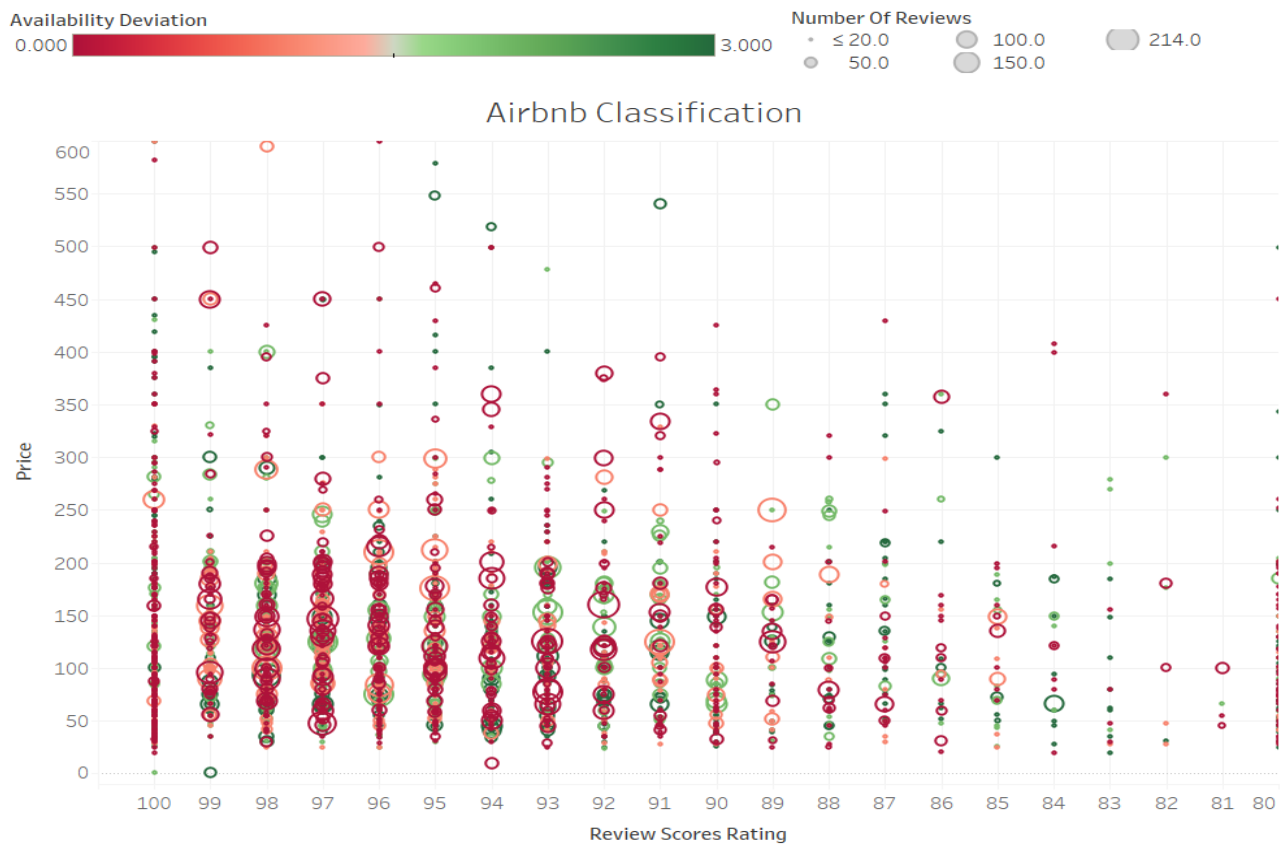
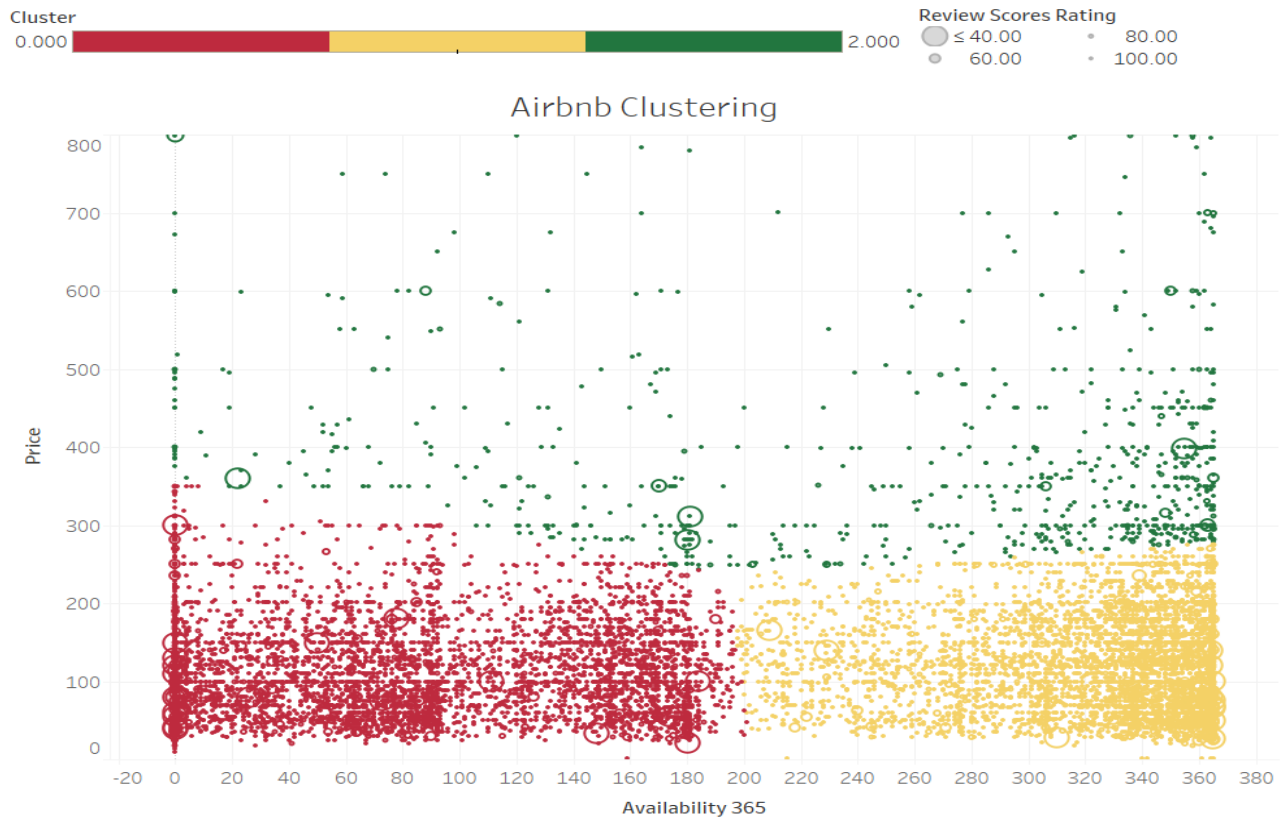
Documents delivered:

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

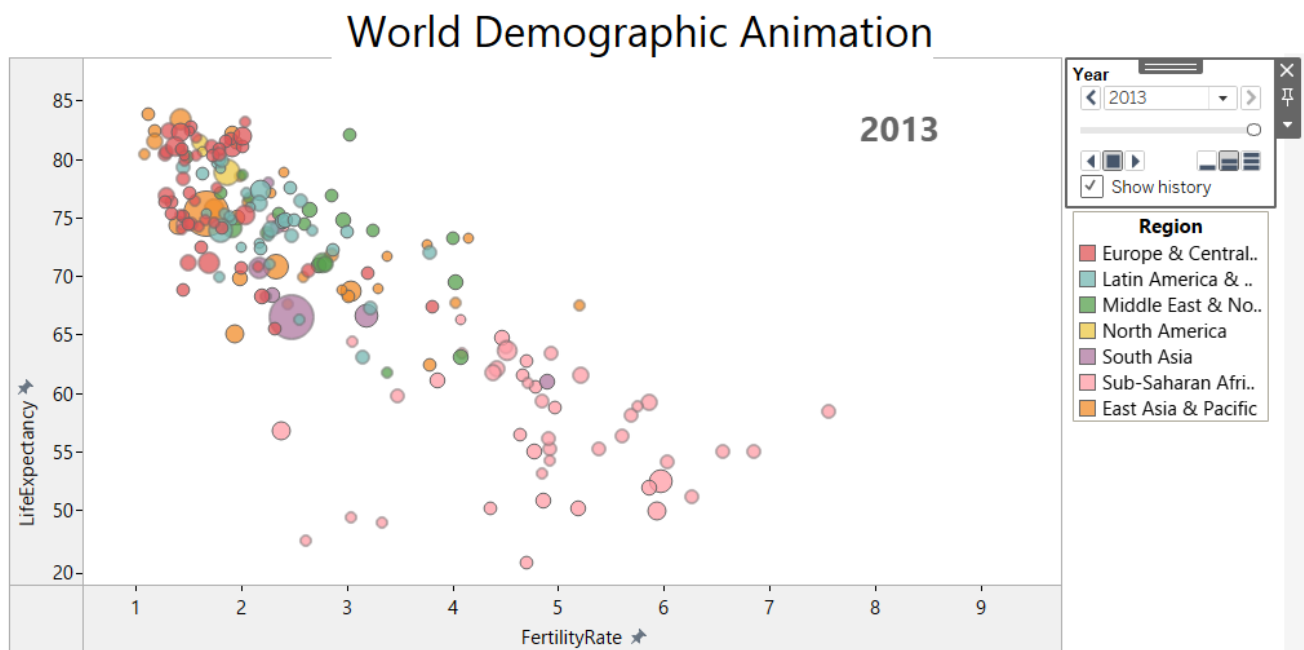
Documents delivered:

Research Report

Duration:

Three Month

Live Data Visualization in Tableau



Client:

Freelance 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

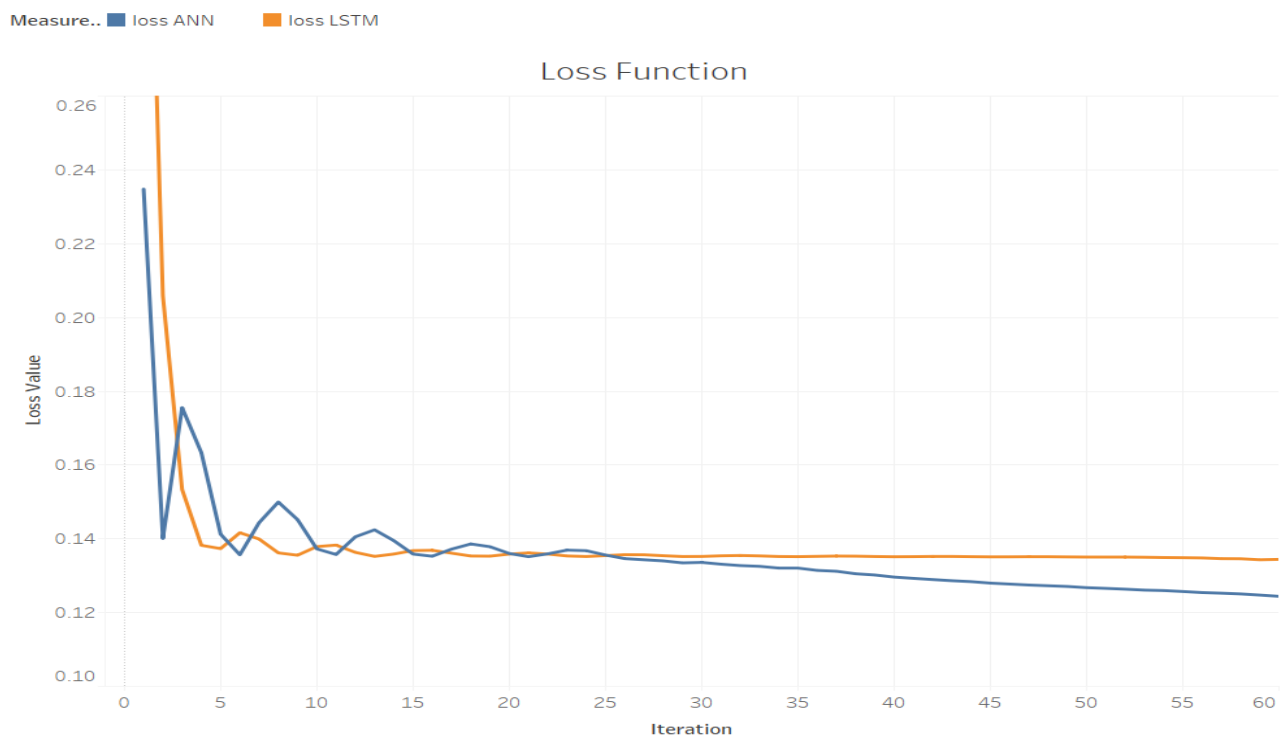
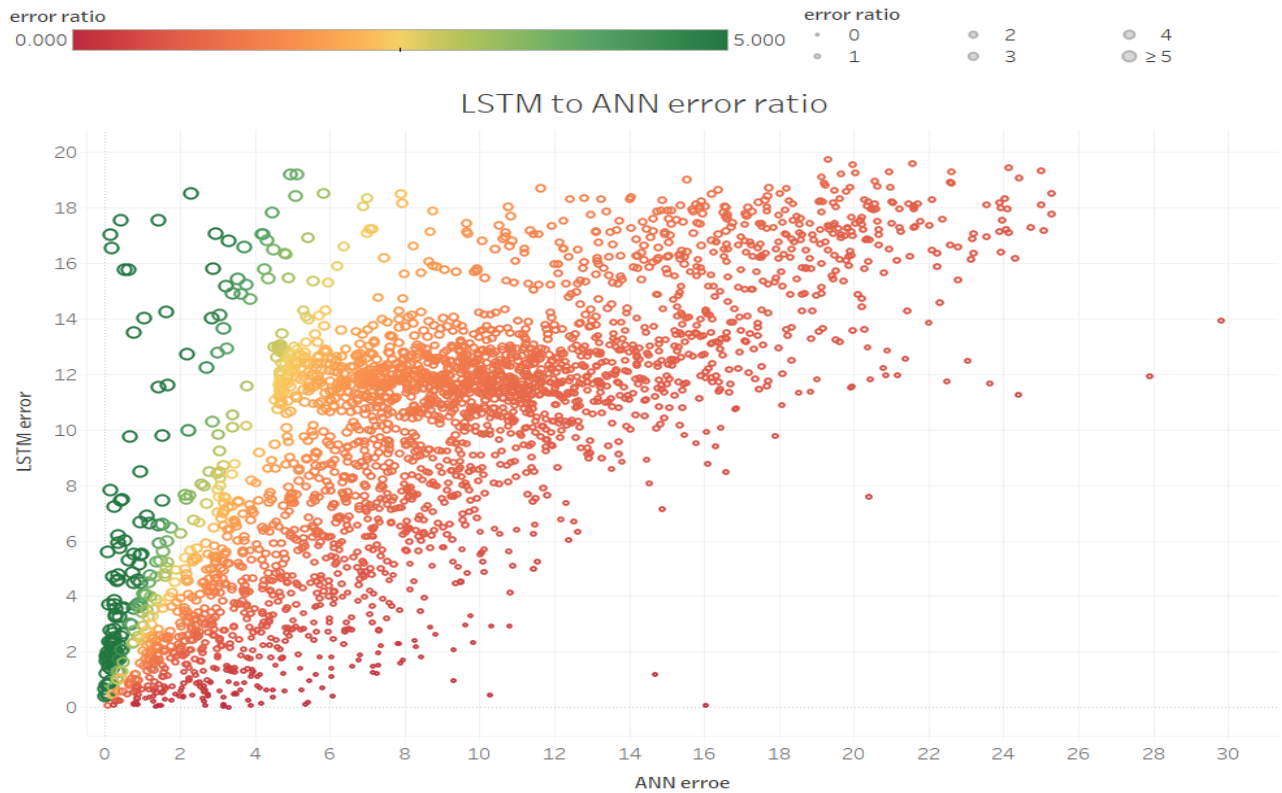
Documents delivered:

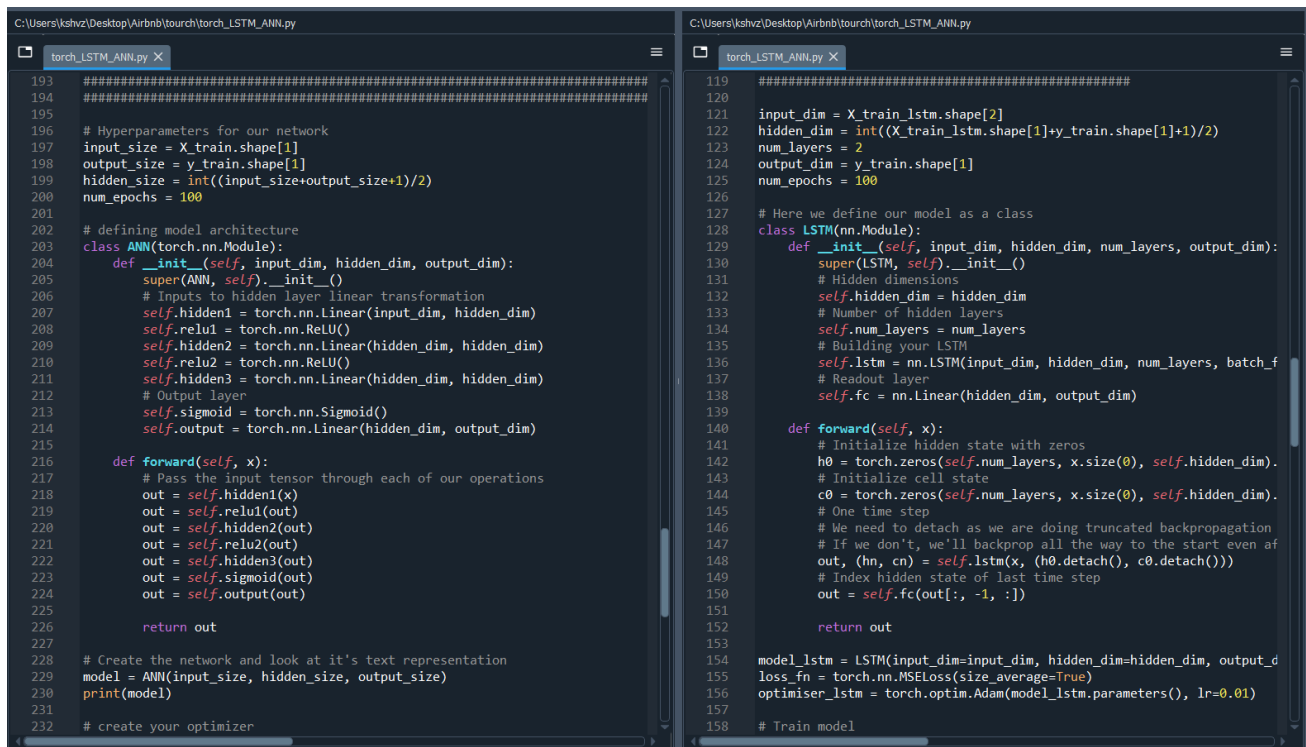
ReadMe file, Animated Video

Duration:

One Month

Comparison between Pytorch LSTM and ANN performance on an inconsistent dataset





```
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
```

```
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

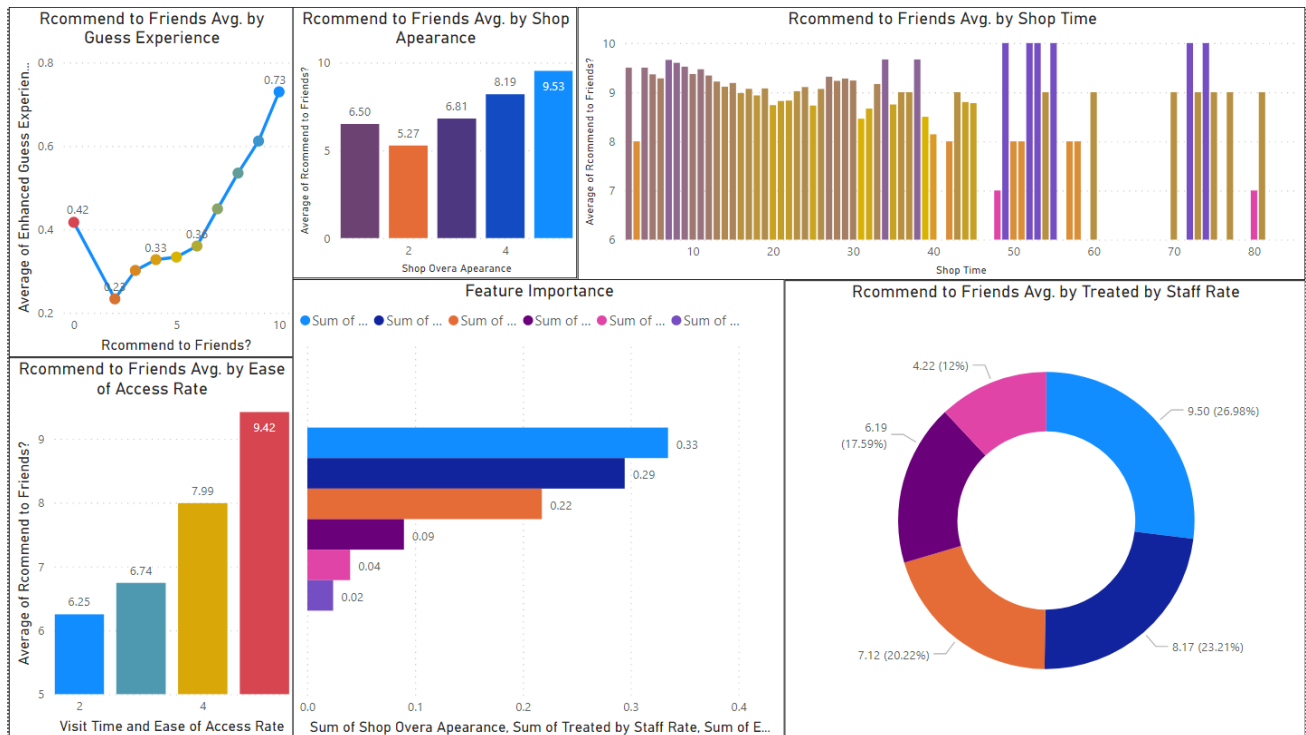
Documents delivered:

Model Description

Duration:

Two weeks

PySpark ML models on survey datasets



```

C:\Users\kshvz\Desktop\pyspark_works\RandomForest\pyspark_randomForest.py
torch_LSTM_ANN.py x pyspark_regression.py x pyspark_randomForest.py x
1 #-*- coding: utf-8 -*-
2 """
3 Created on Sat Oct 30 18:45:34 2021
4
5 @author: kshvz
6 """
7
8 import pandas as pd
9
10 # Finds Spark on local machine
11 import findspark
12 findspark.init()
13
14 # Starts spark session
15 from pyspark.sql.functions import col
16 spark = SparkSession.builder.appName("DataFrame").getOrCreate()
17
18 # Reads dataframe
19 df = spark.read.csv("dataframe.csv", header=False)
20 df.toPandas()
21 print((df.count(), len(df.columns)))
22
23 df_properties = df.describe().toPandas()
24
25 # Select Columns for the model
26 from pyspark.sql.functions import col
27 dataset = df.select(col('_c19').cast('int'),\
28                    col('_c42').cast('int'),\
29                    col('_c43').cast('int'),\
30                    col('_c67').cast('int'),\
31                    col('_c92').cast('int'),\
32                    col('_c227').cast('float'),\
33                    col('_c287').cast('int'))
34
35 dataset.show(n=12)
36 dataset_properties = dataset.describe().toPandas()
37
38 from pyspark.sql.functions import isnull, when, count
39 # Counts the number of null cells on each column
40 dataset.select([count(when(isnull(c), c)).alias(c) for c in
C:\Users\kshvz\Desktop\pyspark_works\RandomForest\pyspark_randomForest.py
torch_LSTM_ANN.py x pyspark_regression.py x pyspark_randomForest.py x
41 # Delete rows having null values
42 dataset = dataset.replace('?', None).dropna(how='any')
43 dataset.select([count(when(isnull(c), c)).alias(c) for c in dataset.columns])
44
45 # Converts 'yes,no' categories to 1,0
46 from pyspark.ml.feature import StringIndexer
47 dataset = StringIndexer(inputCol='_c42', outputCol='n_c42', handleMissing='ignore').fit(dataset).transform(dataset)
48 dataset.show(n=12)
49 dataset.dtypes
50
51 # Drop unnecessary columns
52 dataset = dataset.drop('n_c42')
53 dataset.dtypes
54
55 # Assemble all the features with VectorAssembler
56 from pyspark.ml.feature import VectorAssembler
57 required_features = ['_c19', '_c67', '_c92', '_c227', '_c287']
58 assembler = VectorAssembler(inputCols=required_features, outputCol='features')
59 transformed_data = assembler.transform(dataset)
60 transformed_data = transformed_data.select(['features', '_c43'])
61 transformed_data.show()
62
63 # Split to train and test set
64 (training_data, test_data) = transformed_data.randomSplit([0.8, 0.2])
65
66 # Define the model
67 from pyspark.ml.classification import RandomForestClassifier
68 rf = RandomForestClassifier(labelCol='_c43', featuresCol='features')
69
70 # Fit the model on training set
71 model = rf.fit(training_data)
72
73 # Predict the test set
74 predictions = model.transform(test_data)
75
76 # Evaluate our model
77 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
78 evaluator = MulticlassClassificationEvaluator(labelCol='_c43', predictionCol='prediction')
79 evaluator.evaluate(predictions)
80

```

Client:

Freelance Client

Description:

Random Forest and Linear Regression models applied on a survey dataset for a chain store in Thailand. Since the dataset was containing over 200 columns and 3000 rows, the PySpark API was used to predict the '*Suggestion to the Friend?*' attribute based on other relevant features. The model achieved the accuracy of %65 and maximum feature importance of %33. Microsoft PowerBI was used to visualize the results.

Technologies used:

PySpark, Microsoft PowerBI

Teamwork:

Individual

Documents delivered:

Model Description

Duration:

Three weeks