تحليل علم البيانات باستخدام python – تطبيق عملي ـ بايثونات تحليل البيانات بلغة البايثون – تطبيق عملي كتبه لبنى الحناكي Last updated يونيو 18, 610 شارك المقال تحليل البيانات بلغة البايثون من المهام التي يسأل عنها العديد من الأشخاص المهتمون في مجال تحليل البيانات. عن المشروع في هذا المشروع، csv". في هذا المشروع، يمكن العثور على وصف المشكلة الأصلي ومجموعة البيانات هنا. في هذا التحليل ، matplotlib inline import numpy as np import matplotlib. pyplot% as plt import seaborn as sns معالجة البيانات في الشيفرات البرمجية (الأكواد) التالية، أريد استكشاف مجموعة البيانات وتقييمها بعمق لفهمها بعمق وإيجاد الإجابة على الأسئلة التفصيلية التالية: كم عدد العينات في مجموعة البيانات؟ كم عدد الأعمدة في مجموعة البيانات؟ ما نوع (أنواع البيانات) في المتغيرات؟ هل هناك حاجة لهندسة البيانات؟ مثل تحويل نوع البيانات أو إنشاء المزيد من الأعمدة ببيانات مفيدة؟ هل هناك أي تكرارات؟ هل هناك أي قيمة مفقودة؟ هل هناك أي صفوف مكررة في مجموعة البيانات كم عدد القيم الفريدة غير الفارغة في مجموعة البيانات؟ # Load and read no show appointments data and 'print out a few lines appointment_df= pd. csv) لفحص أنواع البيانات والبحث عن حالات البيانات المفقودة أو التي يحتمل أن تكون خاطئ، تساعد وظيفة الوصف describe في الحصول على نظرة عامة على البيانات من وجهة نظر الملخص الإحصائي. هذا مفيد أيضًا لاكتشاف أي أخطاء محتملة قد تحتاج إلى نظرة فاحصة. # Find (Create the summary report describe (any anomalies in the data)() كم عدد الأعمدة في مجموعة البيانات؟ appointment_df. ScheduledDay 110527 non-null object AppointmentDay 110527 non-null object Diabetes 0 Alcoholism 0 Handcap 0 SMS_received 0 No-show 0 appointment_df. columns Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay', 'AppointmentDay', 'Age', 'Scholarship', 'Hipertension', 'Handcap', 'SMS_received', 'No-show'], dtype='object') appointment_df. dtypes PatientId float64 AppointmentID int64 Gender object AppointmentDay object Neighbourhood object Hipertension int64 SMS_received int64 # checking all possible values on each columns print(appointment_df. Age. Neighbourhood. print(appointment_df. Scholarship. unique()) print(appointment_df. unique()) print(appointment_df. print(appointment_df. Alcoholism. unique()) print(appointment_df. print(appointment_df. SMS_received. print(appointment_df['No-show']). لفحص اذا كان هناك سجلات مُكررة أم لا نستخدم السطر التالي: حيث لا يُوجد سجلات مكررة في البيانات. لا تحتوي مجموعة البيانات على أي قيم مفقودة (NaNs). لن نتمكن من تحليل ما إذا كانت ساعة الموعد لها علاقة بعدم الحضور. تنظيف البيانات الحد الأدنى للعمر هو ـ1 ، ومن الواضح أن الأشخاص لا يمكن أن يكون لديهم سن ـ1. وكذلك لتحسين إمكانية إعادة تعديل قاعدة البيانات. يجب تغيير نوع العمودين Schedule_day وointment_day إلى datetime. appointment_df. query('Age == -1') ## show all recordes that meet the condiction appointment_df. loc[appointment_df. print("Unique Values in `Age` => {}". format(np. وستكون النتيجة كما يلي: 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 102 115] # Drop 'PatientId' and 'AppointmentID' from appointment dataframe appointment_df. appointment_df. RangeIndex: 110527 entries, 0 to 110526 Data columns (total 12 columns): Gender 110527 non-null object ScheduledDay 110527 non-null object Neighbourhood 110527 non-null object Scholarship 110527 non-null int64 dtypes: int64(7), inplace=True) rename(columns= lambda x: x. lower(), appointment_df. head(2) handicap=appointment_df. map({0:0, 1}) to_datetime(appointment_df['scheduledday']). date. appointment_df['appointmentday'] = pd. to_datetime(appointment_df['appointmentday']). dt. date. astype('datetime64[ns]') # Check if the type is now datetime appointment_df. # Print Unique Values for 'AppointmentDay' print("Unique Values in Appointment Day are: {}". Unique Values in Appointment Day are: ['2016-04-29' '2016-05-02' '2016-05-03' '2016-05-04' '2016-05-05' '2016-05-13' '2016-05-14'

2016-06-08' '2016-06-07' '2016-05-18' '2016-05-17' '2016-05-16"] يمتد يوم الموعد ما يزيد قليلاً عن شهر واحد على عكس اليوم المجدول الذي يمتد حوالي 7 أشهر. العمليات الاستكشافية على البيانات بعد تنظيف مجموعة البيانات، # and plot basic histogram charts appointment_df. hist(figsize=(15, 8)); من بين 110500 مريض ، يأتي حوالي 88000 (80٪) من المرضى للزيارة بعد الموعد وحوالي 20000 (20٪) يتخطون مواعيدهم. و "نعم" إذا لم يحضر appointment_df. Gender. value_counts(). plot(kind='bar') #show number of female and male, we can observed that female is double male ax = sns. countplot(x=appointment_df. gender, noshow, data=appointment_df) ax. set_title("Show/NoShow for Females and Males") x_ticks_labels=['Female', ax. set_xticklabels(x_ticks_labels) plt. xlabel('Gender') plt. ylabel('Number'); plt. show() لذلك، قد يكون الجنس عاملاً مهمًا. ولكن إذا نظرنا عن كثب إلى توزيع NoShow عبر Male's و Female فهو متماثل تقريبًا. قد لا يلعب الجنس دورًا مهمًا في تحديد ما إذا كان المريض يأتي في زيارة أم لا. # Use this, and more code cells, to explore your data. Don't forget to add 2)) age, أيضًا، من ناحية أخرى، يبدو أن الفئة العمرية من 40 إلى 60 قد أبدت بالفعل اهتمامًا بالموعد والحضور عند مقارنتها بالفئات العمرية من 0 إلى 20. فإن مجموعة البيانات هذه غير متوازنة،