**פקודות הערות וסיכומים מבוא לניתוח נתונים פייתון**

|  |  |
| --- | --- |
| **vacc\_df.shape** | **נותן מספר שורות ועמודות** |
| **vacc\_df.info()** | **נותן מידע על הדטה שלנו, איזה עמודות ,כמה ערכים חסרים בכל עמודה ומה סוג הטייפ של העמודה** |
| **vacc\_df.columns** | **נותן לנו רשימה עם שמות העמודות וסוג** |
| **vacc\_df.head()** | **נותן דטה מהתחלה לפי מספר השורות שנזין לה בסוגריים** |
| **vacc\_df.describe()** | **מציג נתונים מתמטים של כל עמודה בדטה :ממוצע,ערך מוחלט ,מינימום ,מקסימום, רבעונים** |
| **Df.tail()** | **דטה מהסוף לפי מספר שורות שנזין** |
| df **=** pd.DataFrame | **אפשר ליצור דטה בעזרת הפקודה: ישר אחרי פותחים סוגרים רגילים ואז מילון, במפתח שמים שם סטרינג, ובערך שמים רשימה עם הערכים. בסוף שמים index עם רשימת השורות** |
| **vacc\_df.location**  **vacc\_df['location']** | **דרכים להיציא דטה של עמודה ספציפית שאנחנו בוחרים, אי אפשר להשתמש בשיטת הנקודה אם יש רווח על השם העמודה** |
| **type(vacc\_df.location)** | **נותן את סוג הטיים של העמודה** |
| **vacc\_df.location[333]** | **נותן את העמודה במקום המסויים** |
| **vacc\_df[['location','date']]** | **כדי להוציא דטה שלנו עם יותר מעמודה אחת צריך לעשות סוגריים כפולים עם שמות העמודות הרצויות** |
| **vacc\_df.iloc[0]**  **vacc\_df.iloc[1:2] # second and third row** | **עושה סלייסינ לפי מספר שורה נותן לך אותה במלואה** |
| **vacc\_df.iloc[:, 0:2]** | **הפסיק מפריד לנו בין שורות לעמודות , נקודתיים נותן לנו כמה אנחנו רצים בשורות או בעמודות, לא כולל מספר אחרון שרשום, אם רוצים הכל צריל נקודותיים בלי מספרים כלל** |
| **vacc\_df.loc[:,'daily\_vaccinations']**  **vacc\_df.loc[:,['location','daily\_vaccinations']]** | **נותן חיתוך לפי שם של עמודה או שורה, הפסיק מפריד ביניהם,כולל מספר אחרון שכתוב** |
| **vacc\_df.loc[vacc\_df.location == 'Israel']** | **נותן את כל הדטה על הערך המסויים שבחרנו לפי העמודה** |
| **two\_countries = vacc\_df.loc[(vacc\_df.location == 'Israel') | (vacc\_df.location == 'Denmark')]** | **בדומה לקודם הוספנו קו ישר בינהם וזה אפשר לנו לקחת עוד ערך** |
| **two\_countries.index.values** | **מוציא את כל מספרי השורות של מה שרצינו לפני** |
| **two\_countries.count()** | **קאונט סופר לנו כמה שורות לכל עמודה, בלי להתחשב בשורות שם נאל** |
| **vacc\_df[vacc\_df['location'].str.contains('World')]** | **התבייטות על ערך מסויים שרוצים, דרך סטרינג,זה לנו את כל מה שמכיל את המילה שכתבנו, גם ערכים אחרים שכוללים אותה.** |
| **vacc\_df\_noWorld = vacc\_df.loc[vacc\_df.location != 'World']** | **מוציא לנו את הדטה ללא ערך שאנחנו בוחרים** |
| **Df.max()** | **נותן לנו את המקום עם הערך גבוהה ביותר מספרית או את הערך האחרון מבחינה אלפאבתית** |
| **vacc\_df\_noWorld.loc[vacc\_df\_noWorld.total\_vaccinations == max\_vacc]** | **אפשר להוציא את כל השורה של הערך המקסימלי שרצינו, בדוגמצא שציינו הערך שהשוואתי אליו הוא משתנה שהוצאתי לפני כערך מקסימלי** |
| **vacc\_df\_noWorld.total\_vaccinations.mean()** | **מוציא ממוצע על עמודה שנבחר** |
| **vacc\_df.isnull().sum()** | **נקבל את כל העמודות עם מספר הערכים החסרים בהם באופן סכום** |
| **vacc\_df['daily\_vaccinations'].notnull().sum()** | **נקבל את כל הערכים הקיימים בעמודה** |
| **vacc\_df['daily\_vaccinations'].isnull().sum()**  **np.isnan(vacc\_df['daily\_vaccinations']).sum()** | **2 הדרכים עושות אותו דבר: מוציאות את מספר הערכים החסרים ל עמודה מסויימת. דרך שניה על ידי ספריית נאמפי** |
| **zimbabwe[zimbabwe['total\_vaccinations'].notnull()]** | **אפשר להוציא את הדטה ללא אחרים חסרים בעזרת הפקודה, לפי בחירה של הדטה שאני רוצה כמובן** |
| **zimbabwe.dropna(subset = ['total\_vaccinations'])** | **דרופאנאי מוריד ערכים חסרים, סאבסט נותן לנו לבחור מאיזה עמודות להוריד** |
| * .dropna(axis=1) - drop columns instead of rows |  |
| **vacc\_df.fillna(0,inplace=True)** | **ממלא ערך חסר, מצד שמאל לפסיק במה נבחר , בצד ימין הגדרנו כאשר יש ערך חסר במקום שווה לאמת** |
| **vacc\_df['newTotal'] = vacc\_df.groupby('location')['total\_vaccinations'].apply(lambda x: x.fillna(method='ffill'))**  **vacc\_df.iloc[52:62,[0,2,3,12]]** | **מייצר דטה חדשה מהדטה הקיימת** |
| **vacc\_df['total\_vaccinations'].interpolate(method ='linear', inplace = True)** | **מייצרכמו מאין סדרה רציפה שבה הוא ממלא את הערכים על ידי מטודה ליינר** |
| **top5Avg = top5Avg.reshape(len(top5Avg),1) # for display purpose**  **print(top5Avg)** | **מחליף את ההדפסה להדפסה בטור על ידי רישייפ והמספר 1** |
| **covid\_df['date'] = pd.to\_datetime(covid\_df.date)** | **לשנות לטייפ דייט טיים כדי לקבל את הדברים לפי יום חודש שנה...** |
| **covid\_df['year'] = pd.DatetimeIndex(covid\_df.date).year**  **covid\_df['month'] = pd.DatetimeIndex(covid\_df.date).month**  **covid\_df['day'] = pd.DatetimeIndex(covid\_df.date).day**  **covid\_df['weekday'] = pd.DatetimeIndex(covid\_df.date).weekday**  **covid\_df** | **הוספת עמודות נוספות לדטה שלנו עם המשתנה של הדייט טיים, נוכל בעזרת זה להשתמש בעמודות האלה בפרט** |
| **# print todays date in pandas:**  **now = pd.to\_datetime(str(date.today()), format='%Y-%m-%d')**  **print("Today's date:")**  **print(now)** | **מדפיס את היום שיש היום** |
| **היסטוגרמות ועוד** |  |
| **gym.plot()** | **מציג פלוט לפי קווים** |
| **df\_new = df\_new.rename(columns={'height\_f': 'height\_m'})** | **החלפת שם לעמודה** |
| **grouped\_df.sort\_values('people\_fully\_vaccinated\_per\_hundred', ascending = False)** | **ממיין מהגדול לקטן לפי מה שנבחר** |
| **grouped\_df.hist(bins= 50)** | **היסטוגרמה :סופר כמה פעמים מופיע כל ערך** |
| std() | **סטיית תקן** |
| **boxplot = df.boxplot(column=['C1', 'C2', 'C3'])** | **קו ירוק נותן חציון, קו כחול עליון, קצה המלבן העליון, רבעון עליון, ההפוך זה רבעון תחתון. נקודות מראות חריגה** |
| **house\_df.plot.scatter(x = 'median\_house\_value', y = 'median\_income')** | **נותן סקאטר, התפלגות להציג את הדטה שבחנו, צריך לבחור איקס ווואי כדי להחליט באיזה הקשר אנחנו רוצים להציג את הדברים מבחינת קשר** |
| **house\_df[['median\_income', 'median\_house\_value']].corr(method='pearson')** | **קור עושה קורלציה, לפי מטודת פירסן** |
| **plt.scatter(house\_df['total\_bedrooms'], house\_df['households'])** | **דרך שונה להציג את ההתפלגות, הפעם דרך מטלאב, מצד שמאל לפסיק אומר את איקס, מצד ימין את וואי** |
| **sns.scatterplot(data=house\_df, x='total\_bedrooms', y='households')** | **אותו דבר הפעם עם סיבורן(נראה הכי טוב)** |
| **features = ['median\_house\_value', 'housing\_median\_age',**  **'median\_income']**  **pd.plotting.scatter\_matrix(house\_df[features])** | **תצוגת מטריצאלית של קולציות** |
| **sns.pairplot(house\_df[features], height = 2.5)** | **אותו דבר עם סיבורן** |
| **correlation\_matrix = house\_df[features].corr()**  **correlation\_matrix.style.background\_gradient(cmap='coolwarm')** | **קורלציה בתצוגת היטמאפ, פאנדס** |
| **features = ['median\_house\_value', 'housing\_median\_age','median\_income','total\_bedrooms','population']**  **correlation\_matrix = house\_df[features].corr().round(2)**  **sns.heatmap(data=correlation\_matrix,cmap='Greens', annot=True)**  **correlation\_matrix** | **קורלציית תצוגת סיבורן** |
|  |  |
| **mrkt\_df.columns = mrkt\_df.columns.str.replace(' ', '')** | **מסדר את הרווחים של העמודות בעזרת ריפלייס** |
| **mrkt\_df['Dt\_Customer'] = pd.to\_datetime(mrkt\_df['Dt\_Customer'])** | **דרך נוספת להחליף לדייט טיים** |
| **mrkt\_df['Income'].plot(kind='box', figsize=(3,4), patch\_artist=True)** | **עוד דרך לעשות בוקספלוט** |
| **#remove binary and ID**  **df\_to\_plot = mrkt\_df.drop(columns=['ID', 'AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'Response', 'Complain'])**  **#numeric**  **df\_to\_plot = df\_to\_plot.select\_dtypes(include=np.number)** | **שני דרכים לקחת את הדטה שלנו ולהפוך אותה רק למספרית במטרה לעשות איתה דברים כמו קורלציות וכו...** |
| **df\_to\_plot.plot(subplots=True, layout=(4,4), kind='box', figsize=(12,14), patch\_artist=True)**  **plt.subplots\_adjust(wspace=0.5);** | **לייצר סאבלוטס, הרבה פלוטים במכה** |
| **mrkt\_df = mrkt\_df[mrkt\_df['Year\_Birth'] > 1900].reset\_index(drop=True)** | **מחיקת אינדקסים לפי תנאי מסויים** |
| **sns.lmplot(x='NumWebVisitsMonth', y='NumWebPurchases', data=mrkt\_df)** |  |
| **pd.pivot\_table(iris\_df,index=['class'],values=['sepal\_length','sepal\_width','petal\_length','petal\_width'])** | **פיבוט טייבל, ברירת מחדל לפי ממוצע, עוש לנו דטה חדשה** |
| **plt.title("Distribution of Species")**  **iris\_df['class'].value\_counts().plot.pie(autopct="%1.1f%%")**  **plt.show()** | **עושים פלוט פאי, עם כותרת** |
| **plt.subplot(2,2,1)** | **כשרוצים לעשות יותר מפלוט אחד** |
| **מודל עץ קלאסיפייר לפי הסדר** |  |
| **model1 = sk.tree.DecisionTreeClassifier(criterion="gini")** | **מגדירים מודל עץ בתור התחלה, הדיפולט זה גיני** |
| **features = ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width']**  **X = iris\_df[features]**  **y = iris\_df['class']** | **מגדירים פיצ'רס ואיקס וואי** |
| **model1.fit(X, y)**  **prediction2 = model2.predict(X\_test)**  **prediction2** | **לאמן את המודל**  **ואחרי זה עושים פרדיקט לאיקס טסט** |
| **model1.feature\_importances\_** | **סיווג המודל לפי חשיבות** |
| **text\_representation = tree.export\_text(model1)**  **print(text\_representation)** | **לצייר עץ בלי עץ(פחות חשוב)**  **לצייר עץ בצורה מקראית** |
| **fig = plt.figure(figsize=(15,10))**  **tree.plot\_tree(model1,**  **feature\_names = features,**  **class\_names = model1.classes\_,**  **filled=True,**  **rounded = True)**  **plt.show()** | **תיבה בה מגדירים פלוט כדי להוציא את העץ, צריך להגדיר בפנים את שם המודל, את הפיצ'רס שלנו, קלאסס זה סוג של פונקציה כל זה פלוס הצגה** |
| **X\_train, X\_test, y\_train, y\_test = sk.model\_selection.train\_test\_split(X, y, test\_size=0.3, random\_state=1)** | **חלוקה לאימון ומבחן ,** |
| **X\_test.head(2) y\_test.head(2)** | **מוציאים סתם כדי לראות(לא חובה)** |
|  |  |
|  |  |
| **italy\_vacc = italy\_vacc[italy\_vacc['daily\_vaccinations'].notna()]** | **מוריד שורות, עם ערך חסר לפי עמודה שהוגדרה , בדוגמא יש שינוי של הדטה הגדולה גם** |
| **data1 = pd.concat([data1,dfsfd],axis=1)**  **Df["column"]=Df["column"].astype(float)**  **opop = insurance.groupby(['smoker','sex']).size().unstack()** | **חיבור בין דטה פריימים**  **לשנות טייפ של עמודה**  **פקודת סייז על גרופבאי,מפרקת עמודה וסוכמת את המשתנים שבה(שימושי)** |

דוגמת לעץ החלטה:ריגרסו נרצה להוציא במודל זה mse

|  |  |
| --- | --- |
| import pandas as pd  import numpy as np  import os  from datetime import date # get python datetime  import matplotlib.pyplot as plt  import seaborn as sns  import sklearn as sk  from sklearn import tree  from sklearn.tree import DecisionTreeClassifier  from sklearn.tree import DecisionTreeRegressor  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import mean\_squared\_error as mse  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.model\_selection import train\_test\_split  from sklearn.tree import DecisionTreeClassifier  from sklearn.metrics import mean\_squared\_error as mse | אימפורטים על ים דברים |
| url = 'https://raw.githubusercontent.com/nlihin/data-analytics/main/datasets/avocado.csv'  data = pd.read\_csv(url)  data.head() | מקור דטה פלוס תצוגהשלו |
| data['type\_cat'] = data['type'].astype('category').cat.codes  data['region\_cat'] = data['region'].astype('category').cat.codes  features = ['Total Bags', 'AveragePrice', 'type\_cat', 'region\_cat']  X = data[features]  y = data['total\_sold']  X\_train, X\_test, y\_train, y\_test = sk.model\_selection.train\_test\_split(X, y, test\_size=0.3, random\_state=42)  X\_train.head()#for the train | בשורות הראשונות עשינו קאטקוד כדי להפוך את הדטה שלנו למספרית ונוחה לעבודה בחלק מהעמודות,(בעזרת קאט קוד יצרנו עמודה חדשה בדטה) לאחר מכן הגדרנו פיצ'רס, הגדרנו x ,y והגדרנו לייבל, שזו התוצאה שאנו מחפשים לבדוק.  לאחר מכן שורת האימון ואחרי זה הצגנו את האימון(לא קריטי אם לא מבקשים) |
| model = DecisionTreeRegressor(random\_state=42)  model.fit(X\_train,y\_train) | הגדרת סוג המודל שבחרנו לעבוד איתו ואימון המודל |
| def eval(x\_test,y\_test,model):  pred = model.predict(x\_test)  print("MSE: {:.3f}".format(mse(pred,y\_test,squared=False)))  print("Depth:",model.get\_depth())  eval(X\_test,y\_test,model) | פונקיית איווליואיישן, שעושה בתוכה את פעולת הפרדיקט על איקס טסט, ומוציאה את mse , צריך בשביל זה את האימפורט המתאים, ככל שmse קטן יותר זה טוב יותר לבסוף הדפסנו את הפונקציה ואת העומק של העץ. |
| model = DecisionTreeRegressor(max\_depth=4,random\_state=42)  model.fit(X\_train,y\_train)  print("Depth:",model.get\_depth())  eval(X\_test,y\_test,model) | ניסינו שוב להגדיר הפעם עם עומק אחר |
| import sklearn.tree as tree  def plot\_tree(tree\_model,feat,size=(15,10)):  fig = plt.figure(figsize=size)  tree.plot\_tree(tree\_model,  feature\_names = feat,  filled=True,  rounded = True)  plt.show  plot\_tree(model,features) | לבסוף יש פה פונקציה שבתוכה יש את פעולה הפלוט להדפסת העץ כלפי המסך |
|  |  |

דוגמת עץ לקלסיפייר, שימוש ב acc   
חשוב לראות שגם פה בהגדרת הפיצ'רים, הם צריכים להיות מספריים ולכן אם צריך לעשות קאט קוד או גט דאמיז עליהם

|  |  |
| --- | --- |
| url = 'https://raw.githubusercontent.com/nlihin/data-analytics/main/datasets/iris.csv'  iris\_df = pd.read\_csv(url) | לקיחת דטה |
| features = ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width']  X = iris\_df[features]  y = iris\_df['class']  X\_train, X\_test, y\_train, y\_test = sk.model\_selection.train\_test\_split(X, y, test\_size=0.3, random\_state=1) | הגזרת פיצ'רים,לייבל, חלוקת הדטה לאימון ומבחן |
| from sklearn.svm import SVC  svm\_model = SVC(random\_state = 0)  svm\_model.fit(X\_train, y\_train)  svm\_pred = svm\_model.predict(X\_test)  sk.metrics.accuracy\_score(svm\_pred, y\_test) | **דרך אחת לייצר חיזוי (לא השתמשנו בה כמעט)** |
| from sklearn import tree  dt = sk.tree.DecisionTreeClassifier(random\_state=0)  dt.fit(X\_train, y\_train)  pred = dt.predict(X\_test)  acc = sk.metrics.accuracy\_score(pred, y\_test)  acc | **דרך לייצור חיזוי: פוקנציה, שמאמנת את הדטה, ועושה בפרדיקט ומוציאה acc ככל שהוא יותר קרוב ל1 ככה יותר טוב**  **לאחר מכן אפשר לנסות לשנות את עומק העץ** |
| def get\_acc(data ,max\_depth, train\_X, test\_X, train\_y, test\_y):  model = sk.tree.DecisionTreeClassifier(criterion="gini", max\_depth=max\_depth, random\_state=0)  model.fit(train\_X, train\_y)  preds\_val = model.predict(test\_X)  acc = sk.metrics.accuracy\_score(preds\_val, test\_y)  return(acc) | דומה למיקום פלוס העומק שנדרש להזין אותו בקריאה לפונקציה |
| fig = plt.figure(figsize=(15,10))  tree.plot\_tree(model1,  feature\_names = features,  class\_names = model1.classes\_,  filled=True,  rounded = True)  plt.show() | הדפסה של העץ. |
|  | פירסון- בערכים מספריים רציפים בעלי התפלגות נורמלית  ספירמן וקנדל- לערכים מדורגים (לא רציפים) ו/או לא מפולגים נורמלית  קנדל עדיף כשיש מדגם קטן של ערכים |
|  |  |

|  |  |
| --- | --- |
| plt.xlim(15,50)  plt.ylim(0,4000) | נותן לנו את האופציה להתמקד בחלק מהתמונה |
| plt.figure(figsize = (10,6)) | מגדיר גול תמונה |
| sex\_splited = pd.get\_dummies(insurance['sex']) | ])#Makes sex numeric as male and female |
| insurance['female'] = insurance['sex']== 'female' | פותח עמודה חדשה לפי הגדרה |
| sns.scatterplot(data=insurance, x='age', y='expenses',hue= "sex",).set\_xscale('log')  plt.gca().set\_yscale("log",basey =2) | שימוש בלוג, אפשר לרשום בפקודה איקס סקייל ו וואי סקייל, דוגמא ראשונה סיבורנ, דוגמא שניה מאטלאב |
| sns.set(style="darkgrid") | קובע סטייל לפני שמציגים את שאר התמונה |
| cor\_plt = data1.unstack() | מסדר את זה בצורה אחרת,שימושי |
| .std() |  |

|  |  |
| --- | --- |
| boxplot = df.boxplot(column=['C1', 'C2', 'C3']) | בוקספלוט |
| sns.countplot(x="Position", data=nba\_Data) | סיבורן קאונטפלוט,סופר כמה פעמים כל ערך מופיע |
| nba\_Data['Position'].value\_counts().plot.pie(autopct="%1.1f%%") | פאי פלוט,אפשר להזין עמודה אחת ופקודה או שלא |
| plt.title("positon") | נותן כותרת |
| insurance['expenses'].hist(bins= 100 , figsize = (15,10)) | היסטוגמיה |
| sns.scatterplot(data=insurance, x='age', y='expenses',hue= "sex",) | סקאטרפלוט:משתמשים כדי להציג דטה עמוסה(נקודות) |
| sns.barplot(x="type" ,y="total\_sold", hue="type",estimator = sum,data=data) |  |
| sns.heatmap(corr, cmap='coolwarm', center=0, annot = True) | היטמאפ על קורלציה,מציג רק מה שמספרי |
| T\_and\_P.plot.hist(bins=10,alpha=0.7) |  |
| sns.countplot(x="Position", data=data) |  |
| plt.hist(data.total\_sold)  plt.gca().set\_yscale("log")  plt.show() | עם לוג |
| fig, axes = plt.subplots(figsize=(20, 5), ncols=3)  sns.boxplot(ax=axes[0], x='parental level of education', y='reading score', data=data)  sns.boxplot(ax=axes[1], x='parental level of education', y='writing score', data=data)  sns.boxplot(ax=axes[2], x='parental level of education', y='math score', data=data) | הצגה של מספר תמונות יחדיו ומיקומים |
| sns.regplot(x='reading score', y='math score', data=data) | רגרסיה עם תצוגת פריסה |
| plt.hist(male['reading score'], alpha=0.4, label='male')  plt.hist(female['reading score'], alpha=0.4, label='female')  plt.legend(loc='upper right') | תצוגת 2 הסטוגרמות עם כותרות, ומקרא |
| house\_df.plot.scatter(x = 'median\_house\_value', y = 'median\_income') | סקאטרפלוט מאטלאב |
| features = ['median\_house\_value', 'housing\_median\_age',  'median\_income']  pd.plotting.scatter\_matrix(house\_df[features]) | סקאטרפלוט פאנדס, מציג מטריצה של פלוטים |
| sns.pairplot(house\_df[features], height = 2.5) | כמו מעל רק של סיבורן,נראה יותר טוב |
| correlation\_matrix = house\_df[features].corr()  correlation\_matrix.style.background\_gradient(cmap='coolwarm') | מציג טבלת קורלציה בצבעים |
| sns.barplot(x="sex", y="survived", hue="class", data=titanic)  plt.show() | מציג את הדטה |
| sns.catplot(x="sex", y="survived", hue="class", kind="bar", data=titanic)  plt.show() |  |
| ax = sns.stripplot(x="day", y="total\_bill", data=tips)  ax = sns.stripplot(x="total\_bill", y="day", data=tips,  **...**  linewidth=1) |  |
| x = pd.Series(data["Total Bags"], name="Total Bags")  ax = sns.distplot(x) | פלוט עם עקומת נורמל |
| fig, axes = plt.subplots(nrows=1, ncols=2) participaints.plot.bar(ax=axes[0],figsize=(10,5))#Amount of Winners axes[0].set\_title("Number of winners in each olympic") axes[1].set\_title("Ratio of winners in each olympic \n(refer to the amount of each gender)") men\_ratio.plot.bar(ax=axes[1],figsize=(10,5))#Ratio of Winners‏ | דרך להוציא 2 סאבפלוט עם כותרות,מקרא |
| x = np.linspace(-10,10,1000)  plt.plot(x,x\*\*2,'--y')  plt.xlabel('x')  plt.ylabel('y=x^2')  plt.title('y=x^2') | עושה מאין סרטוט של קו ,זו דוגמא לפונקציה |
| .reset\_index() |  |
|  |  |
| model.feature\_importances\_  feature\_importances = pd.DataFrame(model.feature\_importances\_,  index = avi.columns[:5],  columns=['importance']).sort\_values('importance',  ascending=False)  feature\_importances.head() |  |
|  |  |