Opdrachten Python voor Data Science

Inhoud

Opdracht 1 – Functie Beschrijvende Statistiek

Maak een functie die van een numpy array enkele statistische kenmerken bepaalt en in een dictionary terug geeft.

Stap 1: Maak een lijst met een aantal willekeurige gehele getallen.
Tip: Gebruik de functie np.random.randint()

Stap 2: Definieer een functie, bv. describe(), die als argument een array aanneemt.

Stap 3: Maak een lege dictionary. Deze wordt in de volgende stap gevuld met verschillende statistische kenmerken.

Stap 4: Bepaal in de functie de volgende berekeningen en neem de berekende waarden op in de dictionary. Bepaal de volgende kenmerken:
- aantal
- minimum en maximum
- som
- gemiddelde, mediaan, mode
- standaard deviatie.
- eerste en derde kwartiel, Q1 en Q3
- interkwartielafstand IQR
- skewness
- kurtosis

Tip: gebruik de scipy.stats module

Stap 5: Geef de dictionary terug als uitkomst van de functie.

Stap 6: Test de functie door de functie aan te roepen de lijst van willekeurige getallen en het resultaat te printen.

Stap 7: Op basis van het interkwartielafstand kunnen de grenzen voor outliers worden bepaald met Tukey's methode. Outliers liggen 1.5 * IQR beneden het eerste kwartiel en boven het derde kwartiel. Voeg ook deze grenzen toe aan de kenmerken die functie bepaald. Geef ook een list van outliers terug.

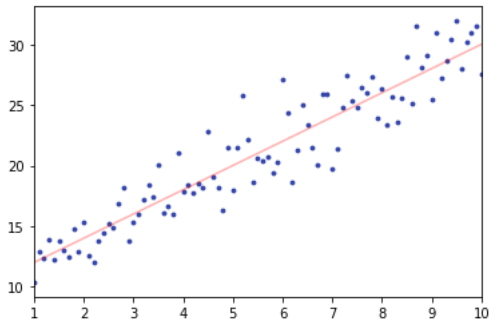
Stap 8: Vergelijk de resultaten met de describe functie uit scipy.stats.

Stap 9: Plot een histogram en een boxplot van de getallen. Plot ook een density kernel plot met seaborn. Tip: Gebruik de functie seaborn.kdeplot()

Stap 10: Gebruik de gemaakte functie om de prijzen van diamanten te beschrijven uit het diamonds dataset. Tip: Deze is te vinden in seaborn: seaborn.loadset('diamonds')

Opdracht 2 – Lineaire Regressie

Lineaire regressie vormt de basis van vele statistische modellen. We gebruiken hiervoor de fictieve data die u zelf in de vorige opdracht heeft gegenereerd. Bepaal de coëfficiënten van de lineaire regressie lijn door een verzameling punten. Anders gezegd: bepaal de coëfficiënten (a en b) van de lijn die het best past bij de punten met vergelijking y = a.x + b.



Stap 1: Genereer de data op basis van een onderliggend model. Definieer hiervoor de vergelijking van het model. Een lineaire functie heeft bijvoorbeeld de vergelijking y= a.x + b. Neem de waarden a = 2 en b = 5 en vervolgens de functie model
met def model(x): return a * x + b of model = lambda x: a * x + b

Stap 2: Voor de gegenereerde data willen we wat minder evenredig verdeelde x-waarden. Definieer hiervoor een aantal uniform verdeelde x-waarden tussen grenzen. Bijvoorbeeld 100 getallen tussen 0 en 10.
Tip: Gebruik de functie np.random.uniform(x_min, x_max, n)

Stap 3: Genereer nu de ruis. Dit zijn net zoveel normaal verdeelde waarden met een gemiddelde van 0 en een te kiezen spreiding (bv. 3).
Tip: Gebruik de functie np.random.randn(n) * spread

Stap 4: Maak nu een array met y waarden. Dit is de waarde van het model voor iedere x de plus ruis.

Stap 5: Plot nu zowel het model als de gegenereerde data in een scatterplot.
Tip: Gebruik plt.scatter(x_data, y_data)

Stap 6: Numpy heeft een module **polynomial** waarmee u een best-fit polynoom kunt bepalen. Gebruik uit deze module de Polynomial class met de method om de best-fit rechte lijn te bepalen. Door 1 als graad op te geven wordt een rechte lijn gebruikt. De return waarde is een tuple met de coördinaten van de vergelijking.
Tip: gebruik de functie np.polynomial.Polynomial.fit()

Stap 7: Plot de dat samen met de lineaire regressie lijn. Neem hiervoor nieuwe evenredig verdeelde x waarden en bepaal de bij behorende y waarden op basis van het model.

Stap 8: Bepaal enkele metrics om de zien hoe goed de fit is. Doe ditz elf met numpy en met de metrics fucties in scikit learn.
Bepaal de
- Root Mean Squared Error RMSE
- Mean Absolute Error MAE
- R-squared

Stap 9: SciPy heeft ook een methode om lineaire regressie uit te voeren en de coördinaten van het lineaire regressie model te vinden. Gebruik deze functie en plot het resultaat. Verifieer dat het hetzelfde is het resultaat van de vorige stap.
Tip: Gebruik scipy.stats.linregress

Stap 10: Herhaal bovenstaande stappen voor andere modellen.
Tip: sinusvormig: model = lambda x: a * np.sin(c*x) + b
kwadratisch: model = lambda x: a*x**2 + b*x + c
exponetieel: model = lambda x: a*b**x + c

Opdracht 3 – Pandas

Stap 1: Load the mpg dataset from seaborn into a dataframe (tip: sns.load_dataset('mpg'))

Stap 2: Explore de dataset with the info() method. Number of observations? Column names? Data types?

Stap 3: Display the distribution of the cars over the origin column.

Stap 4: Seperate the car name in a make and a model. Display the distribution of the makes. Fix the typos.

Stap 5: Plot scatter plots for all pairs of columns with seaborn.pairplot(data=df, hue='origin').

Stap 6: Which 5 cars have the lowest mpg? And the highest 5? Convert mpg to a new column kml.

Stap 7: Convert the weight to a new categorical column with categories 'light', 'medium' and 'heavy'.

Stap 8: Create a cross table with origin versus the calculated weight class with df.pivot_table().

Opdracht 4 – KNMI daggegevens

De KNMI stelt vele datasets beschikbaar m.b.t. het weer en klimaat in Nederland. Onder andere biedt de KNMI gegevens over het weer per dag van 35 verschillende weerstations stations op https://www.knmi.nl/nederland-nu/klimatologie/daggegevens. Vergelijk de temperatuurgegevens van het afgelopen jaar van twee verschillende weerstations in Nederland. Waar is het kouder? Laat met pandas zien of tussen deze stations sprake is van een temperatuurverschil.

Stap 1: Importeer pandas, numpy en matplotlib zoals al eerder gedaan.

Stap 2: Ga naar de bovengenoemde site en download de gegevens van twee verschillende stations. Het is handig om twee station te kiezen die enigszins uit elkaar liggen. Pak de verkregen zip-bestanden uit en plaats de bestanden in de datasets directory. Bijvoorbeeld: De Bilt (260) en Hoogeveen (279).

Stap 3: Bekijk de bestanden met een teksteditor. Bestudeer de opbouw van het bestand. Op welke regel beginnen de daggegevens? Op welke rij staan de kolomheaders?

Stap 6: Lees de gegevens in twee Pandas dataframes met read_csv(). Gebruik het skip argument om de regels aan het begin van het bestand over te slaan.

Stap 7: Maak een selectie van de gewenste rijen op basis van het jaar (bv. 2020). Selecteer het kolom TG om mee te werken.

Stap 8: En voeg beide dataframes onder elkaar samen. Tip: gebruik pd.concat()

Stap 9: De temperatuurwaarden zijn gegeven in 0.1 graden celcius. Converteer de temperaturen naar numerieke waarden en vervolgens naar graden celcius door de waarden te delen door 10. Plaats in een nieuw kolom: GEMIDDELDE.

Stap 10: Converteer het datumveld YYYYMMDD naar een datetime waarde in een nieuw kolom, bv. DATUM. Tip: Gebruik hiervoor pd.to_datetime().

Stap 11: Converteer het STN kolom naar de stationnamen. Gebruik hiervoor de methode map met een dict.

Stap 12: Drop de oorspronkelijke kolommen. Die zijn niet meer nodig.

Stap 13: Pivot de dataframe naar een vorm met de datum als rij index en de verschillende temperatuurmetingen in de kolommen zoals hieronder weergegeven.

	GEMIDDELDE		MINIMUM		MAXIMUM	
STATION	De Bilt	Hoogeveen	De Bilt	Hoogeveen	De Bilt	Hoogeveen
DATUM						
2020-01-01	0.8	0.2	-0.2	-0.7	1.8	1.8
2020-01-02	3.9	1.8	1.3	-0.8	7.4	6.0
2020-01-03	7.6	6.7	4.3	3.3	9.9	9.1
2020-01-04	6.7	5.3	4.0	1.8	8.1	8.1
2020-01-05	6.9	5.4	5.9	1.2	7.6	7.0
...	...	...	...	...	...	...
2020-12-27	5.0	4.4	3.2	3.4	5.9	5.3
2020-12-28	3.6	3.5	1.3	1.4	5.4	5.8
2020-12-29	3.7	3.2	2.5	2.3	5.8	4.2
2020-12-30	4.2	3.1	2.4	1.2	6.8	5.8
2020-12-31	2.1	1.9	-2.1	-1.6	4.0	3.8

366 rows × 6 columns

Stap 14: Plot de beide temperaturen in één plot. Is er een conclusie te trekken?

Stap 16: Maak boxplots van de temperatuur kolommen. Kan hieruit het een en ander worden afgeleid m.b.t. het temperatuur verschil?

Stap 17: Maak een scatterplot van de twee temperaturen en plot ook een lijn waarbij de temperaturen identiek zouden zijn. Is er een conclusie te trekken?

Stap 18: Maak een extra series met het verschil tussen de temperaturen.

Stap 19: Toon een histogram van het temperatuurverschil. Is er een conclusie te trekken?

Stap 20: Plot de temperatuurverschillen. Plot ook een nullijn.

Stap 21: Bepaal het aantal dagen dat het kouder is in Hoogeveen dan in De Bilt.

★ Stap 22: Tot nu toe trekken we conclusies op basis van visualisaties. Met een statistische test kan een conclusie met een zeker waarschijnlijkheid worden aangenomen of verworpen. Hiervoor wordt een Student t-test. Als de p-waarde kleiner is dan 0.01 dan kan met 99% zekerheid worden gezegd dat het kouder is in Hoogeveen.
Tip: Gebruik scipy.stats.ttest_1samp()

Opdracht 5 – Matplotlib

De Gapminders worden beschouwd als een uitstekende voorbeeld van hoe meerdere variabelen in één plot kunnen worden weergegeven. In deze opdracht gaan we deze plots namaken.

Stap 1: Gebruik de World Development Indicators van de World Bank databank.

Stap 2: Werk deze gegevens om naar een dataframe op basis waarvan je een scatterplot kan maken. Neem hiervoor een selectie van de rijen op basis van het gewenste jaar en de gewenste variabelen die u tegen elkaar wilt weergeven. Bijvoorbeeld: GDP per inwoner versus het gezinsgrootte

Stap 3: Maak met matplotlib een scatter plot die zoveel mogelijk lijkt op de Gapminder plots. Denk aan de volgende kenmerken:
- de x en y coordinaten
- de marker kleur
- de grootte van marker
- de titel en de as labels
- de grid
- het gekozen jaar als tekst annotatie

Opdracht 6 – Geopandas

Geografisch gerelateerde data kan worden weergeven op een kaart met de package geopandas.

Stap 1: Installeer de package geopandas als dit nog niet is.

Stap 3: Er zijn vele shape bestanden beschikbaar die door geopandas kunnen worden gebruikt om kaarten weer te geven. Het CBS heeft bijvoorbeeld een bestand voor buurten, wijken en gemeentes die te vinden in de wijk-en-buurtkaart bestanden. Ga naar https://www.cbs.nl/nl-nl/reeksen/redactioneel/geografische-data en download een recente Wijk-en-buurtkaart bestand.

Stap 4: Lees de wijk- en buurtkaart in met geopandas.

Stap 5: Bestudeer de gegevens die beschikbaar zijn in de dataframe. Bekijk hierbij ook de toelichting in de bijgegeven toelichting op de site.

Stap 6: Plot de kaart van Nederland. Laat hierbij de WATER regio's buiten beschouwing.

Stap 7: Kies een grootheid en geef die in de kaart weer. Bijvoorbeeld de bevolkingsdichtheid per gemeente.

Stap 8: Bepaal ook de verhouding tussen het aantal mannen en vrouwen per gemeente en geef dit weer met geopandas. Wat valt op?

Opdracht 7 – Data Preparatie

Voor de classificatie worden gebruik gemaakt van de penguins dataset. Voordat we de classificatie model kunnen bepalen zal de data eerst moeten worden voorbereid.

Stap 1: Lees de data in van seaborn.

Stap 2: Doe een EDA

Stap3: Verwijder rijen met ontbrekende waarden

Stap 4: Bepaal de target en de feature kolommen en split dit in twee dataframes

Stap 5: Encode de categoriale features met on-hot encoding. Vergeet niet rekening te houden met de dummy trap. Tip: gebruik de functie pd.getdummies()

Stap 6: Split de data in een training dataset en een test dataset en sla het resultaat op in vier verschillende dataframes

Stap 7: Schaal de numerieke features naar waarden van dezelfde orde van grootte. Doe dit voor zowel de trainingsdataset als de test dataset. Sla resultaten op in aparte dataframes.
Tip: Gebruik uit de module sklearn.preprocessing de MinMaxScaler of de StandardScaler.

Nu is de data preprocessing klaar en kan worden begonnen met het classificeren. Als het goed is heb je de volgende 6 dataframes:

df_features_train
df_features_test
df_target_train
df_target_test
df_features_train_scaled
df_features_test_scaled

Hier gaan we in de volgende opdracht meet verder.

Opdracht 8 – Classificatie

Deze opdracht gaat over het classificeren van verschillende soorten pinguïns op basis van verschillende kenmerken zoals gewicht, de lengte van de flippers en de lengte en diepte van de snavel. In de opdracht worden de classificatie algoritmes gebruikt en worden de resultaten geëvalueerd.

We gaan nu verder met de voorbereidde datasets uit de vorige opdracht.

Stap 1: Instantieer een **k-Nearest Neighbor** classificatie model. Tip: Gebruik de classifier sklearn.neighbors.KNeighborsClassifier

Stap 2: Train de classifier met de geschaalde trainingsdataset. Tip: Gebruik de fit methode.

Stap 3: Voorspel met de getrainde classifier de target op basis van de geschaalde test features dataset. Tip Gebruik hiervoor de predict methode

Stap 4: Vergelijk de verkregen predictions met werkelijke target in de test dataset. Tel het aantal correcte predictions en bepaal op basis hiervan de accuracy.

Stap 5: Voer de stappen 1 t/m 4 opnieuw uit met een **k-Nearest Neighbor** classificatie algoritme maar gebruik nu de niet geschaalde features.

Stap 6: Voer de stappen 1 t/m 4 uit met een **Naive Bayes** classificatie algoritme. Tip: Gebruik sklearn.naive_bayes.GausianNB()

Stap 7: Voer de stappen 1 t/m 4 uit met een **Decision Tree** classificatie algoritme. Tip: Gebruik sklearn.tree.DecisionTreeClassifier()

Stap 8: Print de Decision Tree. Tip: Gebruik de methode print_tree()

Stap 9: Plot de Decision Tree. Tip: Gebruik de functie sklearn.tree import plot_tree()

Stap 10: Bepaald de bijdrage van de features. Tip: Gebruik

Stap 11: Voer de stappen 1 t/m 4 uit met een **Random Forest** classificatie algoritme. Tip: Gebruik sklearn.ensemble.RandomForestClassifier()

Stap 12: Bepaald nu opnieuw de bijdrage van de features.

Stap 13: Voer de stappen 1 t/m 4 uit met een **Logistic Regression** Classificatie algoritme. Tip: Gebruik sklearn.linear_model.LogisticRegression()

Stap 14: Voer de stappen 1 t/m 4 uit met een **Support Vector Machine** Classificatie algoritme. Tip: Gebruik sklearn.svm.SVC()

Stap 15: Voer de stappen 1 t/m 4 uit met een **Neural Network** Classificatie algoritme. Tip: Gebruik sklearn.neural_network.MLPClassifier()

Stap 16: Bepaal van het bovenstaand classifiers de confusion matrix. Tip: Gebruik **sklearn.metrics.confusion_matrix**

Stap 17: Bepaal vervolgens ook een aantal preformance metrics zoals de accuracy, precision, recall en de F1 score. Tip: Gebruik sklearn.metrics.precision_recall_fscore_support().