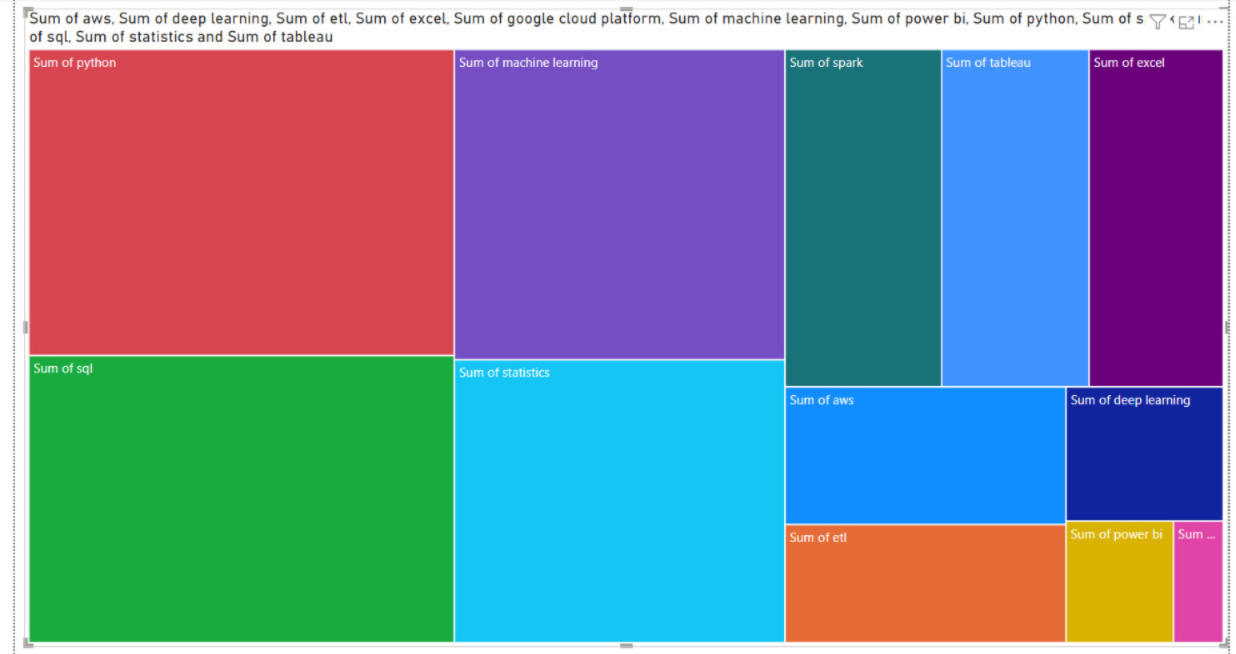


* Top 3 job roles analysis with their jobs count in different sectors and locations, average salary and the distribution of python, ML, AI, Deep learning distribution among them:



* This report analyses the distribution of key skills for different job titles:

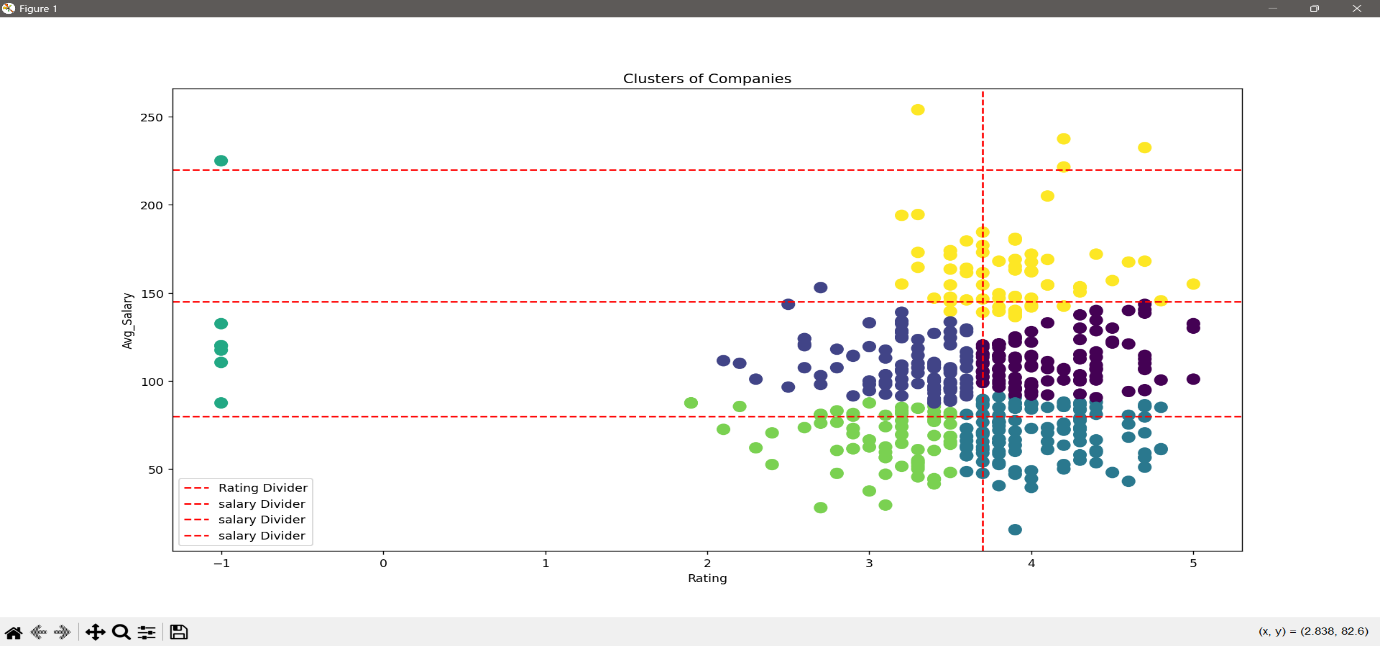
.

CLUSTERING:

Data is clustered with features of average salary and rating:

Insight:

Salary is not dependent on rating.



INSIGHTS:

We have got 6 clusters according to salary and rating (colours):

* we have low salary in high rating, high salary in low rating and

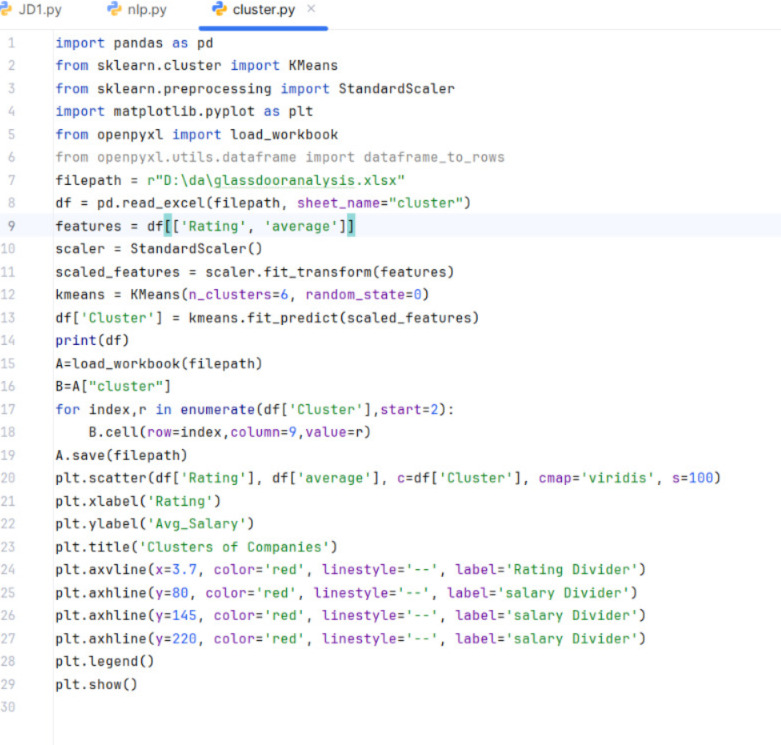
mid-range salary for both low and high rating.

* And that line gives still clear division

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import pandas as pd  
from sklearn.cluster import KMeans  
from sklearn.preprocessing import StandardScaler  
import matplotlib.pyplot as plt  
from openpyxl import load\_workbook  
from openpyxl.utils.dataframe import dataframe\_to\_rows  
#loading file  
filepath = r"D:\da\glassdooranalysis.xlsx"  
df = pd.read\_excel(filepath, sheet\_name="cluster")  
#features for clustering  
features = df[['Rating', 'average']]  
scaler = StandardScaler()  
scaled\_features = scaler.fit\_transform(features)  
  
kmeans = KMeans(n\_clusters=6, random\_state=0)  
df['Cluster'] = kmeans.fit\_predict(scaled\_features)  
  
# Visualize clusters  
plt.scatter(df['Rating'], df['average'], c=df['Cluster'], cmap='viridis', s=100)  
plt.xlabel('Rating')  
plt.ylabel('Avg\_Salary')  
plt.title('Clusters of Companies')  
  
# Add a dividing line :  
plt.axvline(x=3.7, color='red', linestyle='--', label='Rating Divider')  
plt.axhline(y=80, color='red', linestyle='--', label='salary Divider')  
plt.axhline(y=145, color='red', linestyle='--', label='salary Divider')  
plt.axhline(y=220, color='red', linestyle='--', label='salary Divider')  
  
  
# Add legend and show plot  
plt.legend()  
plt.show()





Report on the Code and Output

Code Description:

* Libraries: The code imports necessary libraries such as `pandas`, `sklearn`, `matplotlib`, and `openpyxl`.
* Feature Selection and Scaling: The code selects the 'Rating' and 'average' columns from the DataFrame and scales these features using `StandardScaler`.

the 'Rating' and 'average' columns from the DataFrame for clustering. It then standardizes these features using StandardScaler, which ensures they have a mean of 0 and a standard deviation of 1, making them suitable for clustering.

* K-Means Clustering: the code applies the K-Means clustering algorithm to the scaled features, specifying 6 clusters. The resulting cluster labels are assigned to a new column 'Cluster' in the DataFrame. The print(df) statement displays the DataFrame with the new cluster assignments.
  + It uses Distance formula: sqrt((x2,x1)pow2(y2,y1)pow2)
* Visualization: The code creates a scatter plot of 'Rating' vs. 'average' salary, colours the points based on their cluster labels, and adds dividing lines at specific values for 'Rating' and 'average' salary. The plot is displayed with a legend.

Output Description:

- The output is a scatter plot showing clusters of companies based on their ratings and average salaries.

- The x-axis represents the 'Rating' and the y-axis represents the 'Avg\_Salary'.

- Different colours represent different clusters.

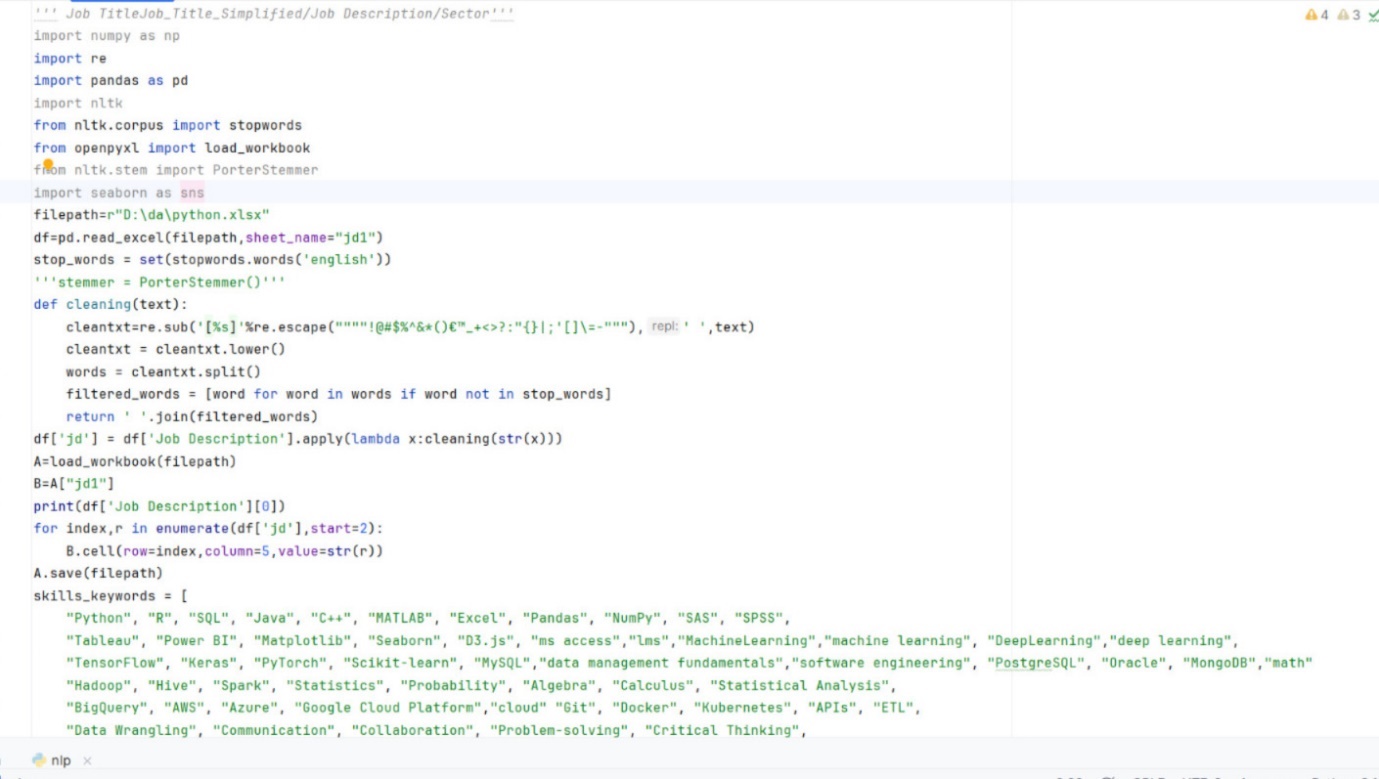
- Red dashed lines are used to divide the plot into sections based on specific 'Rating' and 'Avg\_Salary' values.

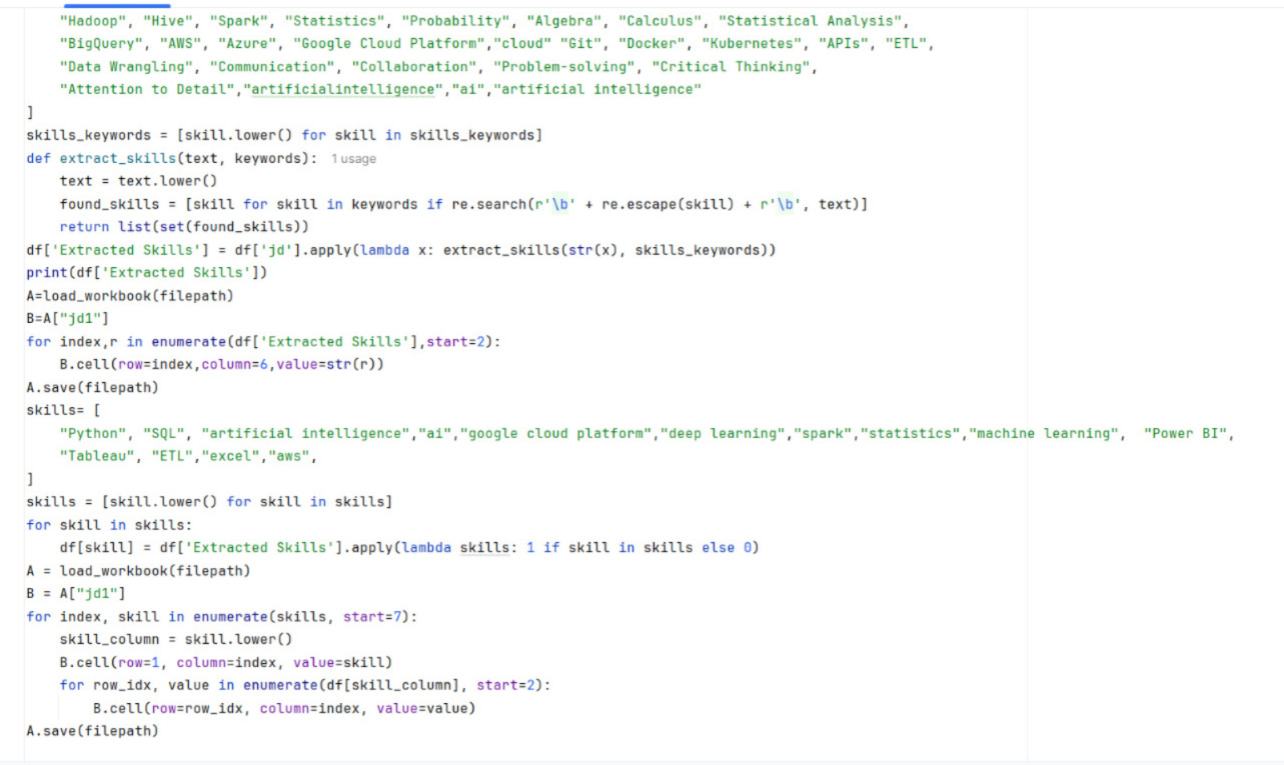
JOD DESCRIPTION TEXT ANALSIS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*''' Job Title/Job\_Title\_Simplified/Job Description/Sector'''*import numpy as np  
import re  
import pandas as pd  
import nltk  
from nltk.corpus import stopwords  
from openpyxl import load\_workbook  
from nltk.stem import PorterStemmer  
import seaborn as sns  
  
  
filepath=r"D:\da\python.xlsx"  
df=pd.read\_excel(filepath,sheet\_name="jd1")  
stop\_words = set(stopwords.words('english'))  
'''stemmer = PorterStemmer()'''

#CLEANING  
def cleaning(text):  
 cleantxt=re.sub('[%s]'%re.escape(""""!@#$%^&\*()€™\_+<>?:"{}|;'[]\=-"""),' ',text)  
 cleantxt = cleantxt.lower()  
 words = cleantxt.split()  
 filtered\_words = [word for word in words if word not in stop\_words]  
 return ' '.join(filtered\_words)  
df['jd'] = df['Job Description'].apply(lambda x:cleaning(str(x)))  
A=load\_workbook(filepath)  
B=A["jd1"]  
print(df['Job Description'][0])  
'''for index,r in enumerate(df['jd'],start=2):  
 B.cell(row=index,column=5,value=str(r))  
A.save(filepath)'''  
skills\_keywords = [  
 "Python", "R", "SQL", "Java", "C++", "MATLAB", "Excel", "Pandas", "NumPy", "SAS", "SPSS",  
 "Tableau", "Power BI", "Matplotlib", "Seaborn", "D3.js", "ms access","lms","MachineLearning","machine learning", "DeepLearning","deep learning",  
 "TensorFlow", "Keras", "PyTorch", "Scikit-learn", "MySQL","data management fundamentals","software engineering", "PostgreSQL", "Oracle", "MongoDB","math"  
 "Hadoop", "Hive", "Spark", "Statistics", "Probability", "Algebra", "Calculus", "Statistical Analysis",  
 "BigQuery", "AWS", "Azure", "Google Cloud Platform","cloud" "Git", "Docker", "Kubernetes", "APIs", "ETL",  
 "Data Wrangling", "Communication", "Collaboration", "Problem-solving", "Critical Thinking",  
 "Attention to Detail","artificialintelligence","ai","artificial intelligence"  
]  
  
# Convert keywords to lowercase  
skills\_keywords = [skill.lower() for skill in skills\_keywords]  
  
  
#function to extract keywords  
def extract\_skills(text, keywords):  
 text = text.lower()  
 found\_skills = [skill for skill in keywords if re.search(r'\b' + re.escape(skill) + r'\b', text)]  
 return list(set(found\_skills))  
  
# Apply the extraction  
df['Extracted Skills'] = df['jd'].apply(lambda x: extract\_skills(str(x), skills\_keywords))  
print(df['Extracted Skills'])  
A=load\_workbook(filepath)  
B=A["jd1"]  
for index,r in enumerate(df['Extracted Skills'],start=2):  
 B.cell(row=index,column=6,value=str(r))  
A.save(filepath)  
skills= [  
 "Python", "SQL", "artificial intelligence","google cloud platform","deep learning","spark","statistics","machine learning", "Power BI",  
 "Tableau", "ETL","excel","aws"  
]  
skills = [skill.lower() for skill in skills]  
  
for skill in skills:  
 df[skill] = df['Extracted Skills'].apply(lambda skills: 1 if skill in skills else 0)  
  
A = load\_workbook(filepath)  
B = A["jd1"]  
  
for index, skill in enumerate(skills, start=7):  
 skill\_column = skill.lower()  
 B.cell(row=1, column=index, value=skill)  
 for row\_idx, value in enumerate(df[skill\_column], start=2):  
 B.cell(row=row\_idx, column=index, value=value)  
  
A.save(filepath)

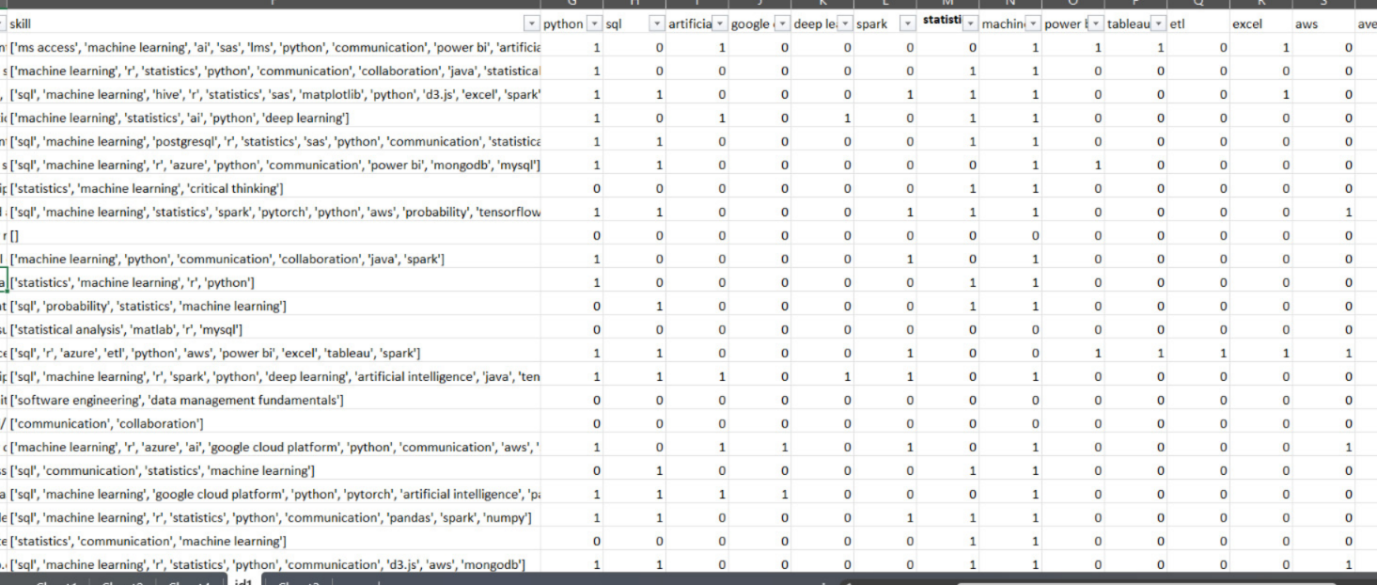
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  




**Cleaning: stop words removal, stemming and special characters removal:**

* **Stop words** are commonly used words in a language that do not carry significant meaning for analysis (e.g., *is, the, and, of, etc.*). These words are usually removed to focus on the content words that matter more.
* **Stemming** is the process of reducing a word to its root or base form by chopping off its suffixes or prefixes. It is often a crude approach that may not always produce actual words, but it helps group similar words together for analysis.
* Key words matching: by having a list of words and pattern matching to get skills
* Get dummies or one hot encoding for skills

EXTRATED SKILL:



Distribution of important skills among job roles

