# ST1 G Capstone Report: Australia Data Science Jobs Dataset


## Pauline Armamento – u3246782


1. **Requirements Analysis:**

   Stage 1: Requirements Specification

   - Appropriate the Australia Data Science Jobs Dataset for Data Analytics, Visualization, Prediction and Deployment.
   - Perform Exploratory Data Analysis and Visualization Task – Identify five problems in the dataset that will be investigated through EDA and presented with visualization techniques.
     1. What is the salary distribution among jobs in the dataset?
     2. Identify the Highest and Lowest Paying Job Titles.
     3. Identify which state offers the most number of Data Science Job Opportunities.
     4. Which is the highest paying State in terms of estimated salary?
     5. Identify which variables is significantly correlated to the Estimate Base Salary
   - Perform Predictive Analytics Task
     - Use a classifier to make predictions on the target variable: Estimate Base Salary
   - Project Implementation and Deployment Task
     - Web app deployment using Streamlit

   Stage 2: System Analysis:

   - **OUTPUT**: Display the results of the identified problems in EDA
   - **PROCESSING**: Use the following processes to obtain output:
     *Problem (1): salary count = Estimate Base Salary (x1 + x2 + … xn)*
     *Problem (2): job title [salary count = Estimate Base Salary (x1 + x2 + … xn)]*
     *Problem (3): data specialist = location [job count]*
     *Problem (4): state [salary count = Estimate Base Salary (x1 + x2 + … xn)/ state[average estame base salary]*
     *Problem (5): correlation = (xi – xbar) (yi – ybar)/ square root((xi – xbar) (yi – ybar))*

   - **INPUT**: Variables needed for the program are the following:

     Problem (1):
     - Estimate Base Salary

     Problem (2):
     - Job Title
     - Estimate Base Salary

     Problem (3):
     - Job Title
     - Data Specialists: Data Scientist, Data Analyst, Data Engineer, Data Consultant, Data Specialist, and Data Manager
     - Estimate Base Salary
     - Location

Problem (4):

- Estimate Base Salary
- State

Problem (5):

- Estimate Base Salary
- Low Estimate
- High Estimate
- Company Rating
- Company Friend Recommendation
- Company CEO Approval
- Company Number of Rater
- Company Career Opportunities
- Compensation and Benefits
- Company Culture and Values
- Company Work Life Balance

**2. Algorithm Design:**

Algorithm design is the process of choosing and creating models to analyze data and make predictions. It's essential because the algorithm selection affects both accuracy and efficiency of a model. The algorithm design for this dataset is shown in Figure 1 below:

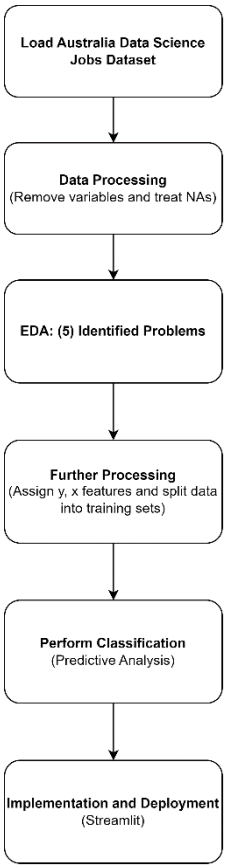



Figure 1: Algorithm Workflow

**Australia Data Science Jobs Dataset**

This job listings data[1] was collected from Glassdoor through web scraping algorithm Selenium (Python). It consists of 2088 observations and 53 variables. It provides valuable information regarding job titles, locations, companies, salary estimates, company ratings and required skills. This data was used to obtain analytical insights in the labor market related to data science in Australia.

**Exploratory Data Analysis**

EDA involves analyzing and summarizing a dataset to gain insights. This includes examining the distribution and relationships of variables through statistical and visualization techniques. Python language, with libraries such as Pandas, NumPy, Scikit-learn, Matplotlib, and Seaborn, was used for data manipulation, cleaning, and visualization on PyCharm and our Streamlit application.

```
# Import Python Packages and read data

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from dataclasses import dataclass
from collections import defaultdict
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

import warnings
warnings.filterwarnings("ignore")


data = pd.read_csv('AustraliaDataScienceJobs.csv')

# Check imported data information and types across columns
# data.info()

# Data Preprocessing

# Drop insignificant columns
data.drop(['Url', 'Company Size', 'Company Founded', 'Company Revenue', 'Job
Descriptions', 'Country'], axis=1, inplace=True)
data.head()

# Check for NAs and missing values
# na_count = data.isna().sum()
# print(na_count)


# We treat the missing values by replacing NA with 0 for numeric_columns.

numeric_columns = ['Estimate Base Salary', 'Low Estimate', 'High Estimate', 'Company
Rating', 'Company Friend Reccomendation', 'Company CEO Approval',
                   'Companny Number of Rater', 'Company Career Opportinities', 'Compensation
and Benefits',
                   'Company Culture and Values', 'Company Senior Management', 'Company Work
Life Balance']

new_datascience_df = pd.DataFrame(data)
new_datascience_df[numeric_columns] = new_datascience_df[numeric_columns].fillna(0)

# After inspecting the dataset, we tried to fill in missing values in the categorial
features:
# Company Type, Company Sector, Company Industry according to the classification of
Company
# Keywords were used to filter the dataset and assign their respective company type,
# sector, and industry.
```

```python
def fill_govt(df, word):
    type_index = df[df['Company'].str.contains(word)].index.tolist()
    df.loc[type_index, 'Company Type'] = "Government"
    df.loc[type_index, 'Company Sector'] = "Government & Public Administration"
    df.loc[type_index, 'Company Industry'] = "National Services & Agencies"
    return df

def fill_private(df, word):
    type_index = df[df['Company'].str.contains(word)].index.tolist()
    df.loc[type_index, 'Company Type'] = "Company - Private"
    df.loc[type_index, 'Company Sector'] = "Management & Consulting"
    df.loc[type_index, 'Company Industry'] = "Business Consulting"
    return df

fill_private(new_datascience_df, 'Pty')
fill_private(new_datascience_df, 'Services')
fill_private(new_datascience_df, 'Group')
fill_govt(new_datascience_df, 'Government')
fill_govt(new_datascience_df, 'Department')
fill_govt(new_datascience_df, 'Council')
fill_govt(new_datascience_df, 'Museum')
fill_govt(new_datascience_df, 'Authority')

def fill_other_company(df, column):
    type_index = df[column].isna()
    df.loc[type_index, 'Company Type'] = "Company - Private"
    df.loc[type_index, 'Company Sector'] = "Other"
    df.loc[type_index, 'Company Industry'] = "Other"
    return df

fill_other_company(new_datascience_df, 'Company Type')
fill_other_company(new_datascience_df, 'Company Sector')
fill_other_company(new_datascience_df, 'Company Industry')


# Remaining NA or missing values were removed from the dataset
new_datascience_df = new_datascience_df.dropna()

# Recheck NA and missing values
# na_datascience_count = new_datascience_df.isna().sum()
# print(na_datascience_count)

# new_datascience_df.info()



# DataScienceJob class was created to present the values in the dataframe
@dataclass(eq=True, frozen=True)
class DataScienceJob:
    job_title: str
    job_location: str
    company: str
    estimate_base_salary: int
    low_estimate: int
    high_estimate: int
    company_type: str
    company_sector: str
    company_industry: str
    company_rating: float
    company_friend_reco: float
    company_ceo_approval: float
    company_number_rate: float
    company_career_opp: float
    compensation_benefits: float
    company_culture_values: float
    company_senior_mgmt: float
    company_worklife_bal: float
    state: str
    python_yn: int
    r_yn: int
    sql_yn: int
```

```
    java_yn: int
    julia_yn: int
    scala_yn: int
    c_yn: int
    cplus_yn: int
    javascript_yn: int
    spark_yn: int
    hadoop_yn: int
    matlab_yn: int
    sas_yn: int
    git_yn: int
    excel_yn: int
    nosql_yn: int
    aws_yn: int
    mongodb_yn: int
    cassandra_yn: int
    hive_yn: int
    bigml_yn: int
    tableau_yn: int
    powerbi_yn: int
    nlp_yn: int
    pytorch_yn: int
    tensorflow_yn: int
    mathematic_yn: int
    statistic_yn: int


data_science_jobs: list[DataScienceJob] = []

for index, row in new_datascience_df.iterrows():
    data_science_jobs.append((row['Job Title'],row['Job Location'],row['Company'],
row['Estimate Base Salary'],
                              row['Low Estimate'], row['High Estimate'], row['Company
Type'], row['Company Sector'],
                              row['Company Industry'], row['Company Rating'],
row['Company Friend Reccomendation'],
                              row['Company CEO Approval'], row['Companny Number of
Rater'], row['Company Career Opportinities'],
                              row['Compensation and Benefits'], row['Company Culture
and Values'], row['Company Senior Management'],
                              row['Company Work Life Balance'], row['State'],
row['python_yn'], row['r_yn'], row['sql_yn'],
                              row['java_yn'], row['julia_yn'], row['scala_yn'],
row['c_yn'], row['c++_yn'], row['javascript_yn'],
                              row['spark_yn'], row['hadoop_yn'], row['matlab_yn'],
row['sas_yn'], row['git_yn'], row['excel_yn'],
                              row['nosql_yn'], row['aws_yn'], row['mongodb_yn'],
row['cassandra_yn'], row['hive_yn'], row['bigml_yn'],
                              row['tableau_yn'], row['powerbi_yn'], row['nlp_yn'],
row['pytorch_yn'], row['tensorflow_yn'],
                              row['mathematic_yn'], row['statistic_yn']))


############################## PART 2: EDA ##############################
# Five questions were created to explore the dataset and perform EDA as follows:

#### 1. What is the salary distribution among jobs in the dataset? ####
# The histogram shows that most number of jobs in the dataset have an
# estimate base salary around $80,000 - $90,000 with very few outliers in the range of
# $200,000 to $300,000. Acquiring such knowledge enables us to comprehend the process
of pay scale
# levels and keeps us aligned with the compensation hierarchy.

# Extract the salaries from the data_science_jobs list
salaries = [job[3] for job in data_science_jobs]

# Create a histogram
plt.figure(1)
sns.histplot(salaries, kde=True)
plt.title('Salary Range and Distribution')
plt.xlabel('Salary')
plt.ylabel('Count')
```

```
# Show the plot
# plt.show()



#### 2. Which Job Titles has the highest estimated salary? ####
# In order to gain insight into the job titles that fall within the upper bracket of
salary distribution,
# a graph displaying the top occupations with the highest estimated base pay in the
dataset was generated.
# This approach allows us to discern which employment positions possess substantial
earning potential
# as previously shown in our first plot.

# Create a dictionary to store total salary and counts for each job title
job_salary = defaultdict(lambda: {'total_salary': 0, 'count': 0})

# Get the salaries for each job title
for job in data_science_jobs:
    job_title = job[0]
    salary = job[3]
    if salary > 0:
        job_salary[job_title]['total_salary'] += salary
        job_salary[job_title]['count'] += 1

# Calculate the average estimated salary
ave_salaries = {}
for job_title, salary_info in job_salary.items():
    if salary_info['count'] > 0:
        ave_salary = salary_info['total_salary'] / salary_info['count']
        ave_salaries[job_title] = ave_salary

# Sort the salary list
sorted_salaries = sorted(ave_salaries.items(), key=lambda x: x[1], reverse=True)

# Display the top 10 job titles from sorted_salaries list
print("Top 10 Job Titles by Average Estimate Base Salary:")
for job_title, ave_salary in sorted_salaries[:10]:
    print(f"{job_title}: ${ave_salary:,.2f}")


# Sort the salary list dictionary
sorted_salaries = {i: j for i, j in sorted(ave_salaries.items(), key=lambda item:
item[1], reverse=True)}

# Get the top 10 job titles by salary
top_job_titles = list(sorted_salaries.keys())[:10]

# Create lists of job titles and mean salaries for the top 10 jobs
top_job_titles_list = []
top_ave_salaries_list = []
for job_title in top_job_titles:
    top_job_titles_list.append(job_title)
    top_ave_salaries_list.append(sorted_salaries[job_title])

# Create a bar graph
plt.figure(2)
plt.bar(top_job_titles_list, top_ave_salaries_list, color='green')
plt.xlabel("Job Title")
plt.xticks(rotation=45, fontsize='7', horizontalalignment='right')
plt.ylabel("Average Estimate Base Salary ($)")
plt.title("Top 10 Data Science Job Titles by Average Estimate Base Salary")

# Show the plot
# plt.show()


#### 3. Determine the location that has the greatest concentration of data specialists
in employment. ####
# For data specialists we enumerated Data Scientist, Data Analyst, Data Engineer, Data
Consultant, Data Specialist,
```

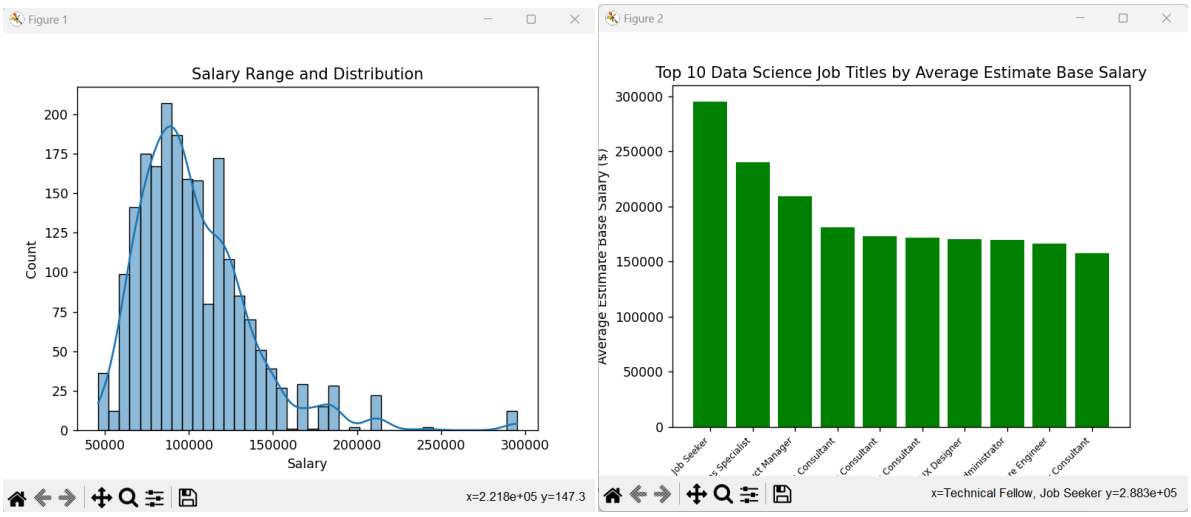
```
# and Data Manager. The research revealed that while Melbourne had the greatest number
of data specialist job vacancies,
# Sydney held the highest volume of such postings. This information offers insight
into where a significant
# proportion of professionals with expertise in data analysis are situated within
Australia.

# Create a dictionary to store count and location for each job role
job_location_count = {'Data Scientist': defaultdict(int), 'Data Analyst':
defaultdict(int), 'Data Engineer': defaultdict(int),
                      'Data Consultant': defaultdict(int), 'Data Specialist':
defaultdict(int), 'Data Manager': defaultdict(int)}

# Get the count of each job role
for job in data_science_jobs:
    job_title = job[0]
    if 'Data Scientist' in job_title:
        job_location_count['Data Scientist'][job[1]] += 1
    elif 'Data Analyst' in job_title:
        job_location_count['Data Analyst'][job[1]] += 1
    elif 'Data Engineer' in job_title:
        job_location_count['Data Engineer'][job[1]] += 1
    elif 'Data Consultant' in job_title:
        job_location_count['Data Consultant'][job[1]] += 1
    elif 'Data Specialist' in job_title:
        job_location_count['Data Specialist'][job[1]] += 1
    elif 'Data Manager' in job_title:
        job_location_count['Data Manager'][job[1]] += 1

# Display the location with the highest count for each job role
for job_role in job_location_count:
    sorted_location = sorted(job_location_count[job_role].items(), key=lambda x: x[1],
reverse=True)
    if sorted_location:
        print(f"{job_role.capitalize()} jobs are highest in {sorted_location[0][0]}
with {sorted_location[0][1]} job postings.")
    else:
        pass


# Create a list for each job role and its highest location count
job_role_location = [(job_role.capitalize(),
sorted(job_location_count[job_role].items(), key=lambda x: x[1], reverse=True)) for
job_role in job_location_count]

# Sort the list by highest location count for each job role
job_role_location.sort(key=lambda x: x[1][0][1], reverse=True)

# Create a list of job roles with highest location count
job_roles = [job_role for job_role, locations in job_role_location]
location_counts = [locations[0][1] for job_role, locations in job_role_location]


# Create a bar graph
plt.figure(3)
plt.barh(job_roles, location_counts, align='center')
plt.xlabel('Count')
plt.ylabel('Job Role')
plt.gca().invert_yaxis()
plt.title('Job Roles by Count')
# plt.show()


#### 4. Which is the highest paying State in terms of estimated salary? ####
# We found that New South Wales offers the highest estimated base salary among all
states, followed by Western Australia
# and Northern Territory. This information provides valuable insights into which
states provide the most significant
# compensation with regards to estimate base salaries.

# Create a dictionary to store total salary and count for each state
state_salary = defaultdict(lambda: {'total_salary': 0, 'count': 0})
```

```
# Get the salaries for each state
for job in data_science_jobs:
    state = job[18]
    salary = job[3]
    if salary > 0:
        state_salary[state]['total_salary'] += salary
        state_salary[state]['count'] += 1

# Calculate the average estimated base salary for each state
ave_salaries = {}
for state, salary_info in state_salary.items():
    if salary_info['count'] > 0:
        ave_salary = salary_info['total_salary'] / salary_info['count']
        ave_salaries[state] = ave_salary

sorted_salaries = sorted(ave_salaries.items(), key=lambda x: x[1], reverse=True)


# Display the Average Estimate Base Salary for each state
print('Average Estimate Base Salary for each state')
for state, ave_salary in sorted_salaries:
    print(f"{state}: ${ave_salary:,.2f}")


# Create a barplot
# sns.barplot(x=list(ave_salaries.keys()), y=list(sorted(ave_salaries.values())),
palette='flare')
plt.figure(4)
sns.barplot(x=list(ave_salaries.keys()), y=list(ave_salaries.values()),
palette='flare')
plt.xlabel('State')
plt.xticks(rotation=45, fontsize='7', horizontalalignment='right')
plt.ylabel('Average Estimated Base Salary ($)')
plt.title('Top High Paying State by Estimate Base Salary')
# Show the plot
# plt.show()



#### 5. Identify which variables are highly correlated to Estimated Base Salary ####
# It was observed that the variables Low Estimate and High Estimate exhibit a positive
correlation with the target
# variable. Conversely, all other variables display a notable negative correlation
with the target variable.

# Calculate the correlation between all the numeric columns
corr = new_datascience_df[numeric_columns].corr()

# Plot the heatmap
plt.figure(5)
plt.rcParams['figure.figsize'] = (10, 7)
sns.heatmap(corr, cmap='PuBuGn', linewidths=.5, annot=True)
plt.title('Correlation Heatmap of Data Science Job Variables')
plt.show()


########################## PART 3: Perform Classification (Predictive Analytics)
##########################
# In the Predictive Analytics component, we have incorporated the variables Low
Estimate and High Estimate with
# high positive correlation as X input features. Additionally, we have also considered
Company Career Opportunities
# a variable exhibiting highest negative correlation to depict an opposite directional
movement.
# By encompassing all these variables together, we can extract supplementary insights
about both target variable and its predictors.

# Assign target variable y
y = new_datascience_df.loc[:, ('Estimate Base Salary')]

# Assign input features X
```

```
X = new_datascience_df.loc[:, ('Low Estimate', 'High Estimate', 'Company Career
Opportinities')]

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Check size of train and test subsets
print('X_train: ', np.shape(X_train))
print('X_test: ', np.shape(X_test))

# print('X_train', X_train)
# print('X_val', X_val)
# print("y_train", y_train)
# print('y_val', y_val)

# Create a random forest classifier object
random_forest = RandomForestClassifier()

# Fit the classifier on the training data
random_forest.fit(X_train, y_train)

# Predict on the test data
y_pred_test = random_forest.predict(X_test)
test_acc = accuracy_score(y_test, y_pred_test)

# Display Summary Statistics
y_pred_test_df = pd.DataFrame(y_pred_test)
summary = y_pred_test_df.describe()
print(summary)

# Display the first 10 predicted values
print('First 10 Predicted Values: ', y_pred_test[:10])

# Display Accuracy Score
print(f'Accuracy Score on Test Set:, {test_acc:.2f}')


# Model Performance Evaluation Metric  - Classification Report
# print(classification_report(y_test, y_pred_test))
```

3. **Evidence of Testing:** The codes were tested by running the program both console and streamlit app. The screenshots below confirm correct output. The Console Application displays figures and console results while the Streamlit Application displays the same result after running the strm.py file.

**(a) Console Results: Figures 1-5, Console Window**



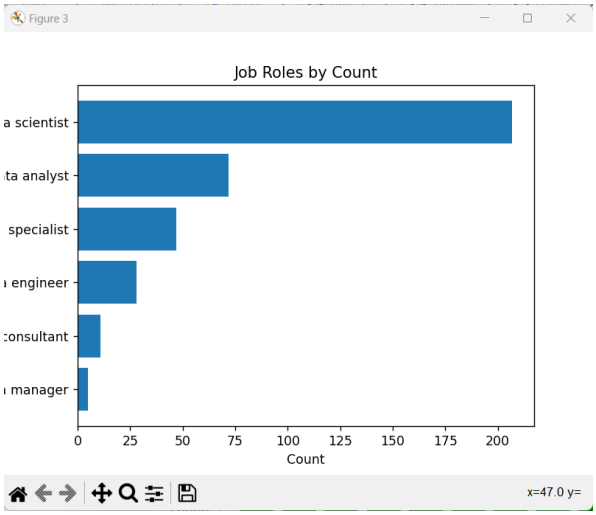
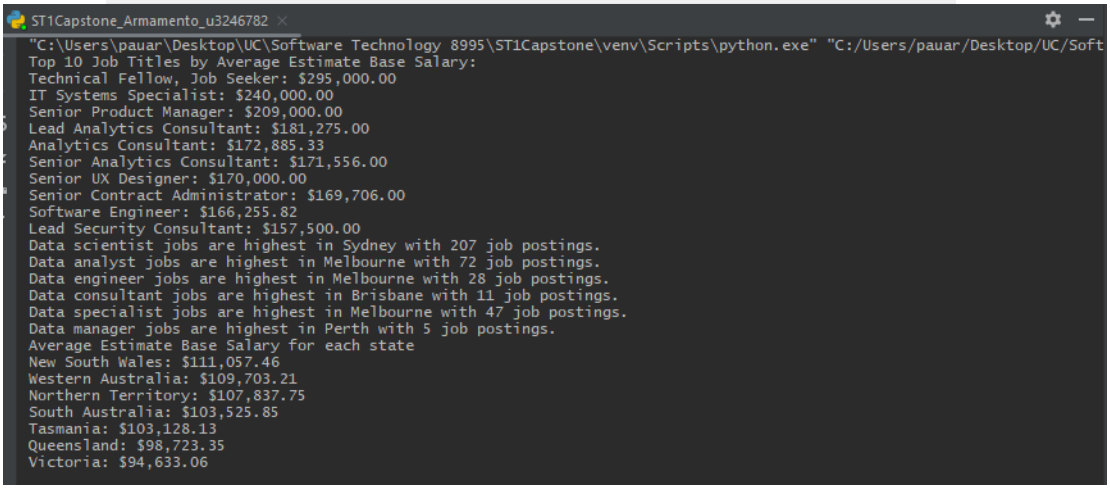


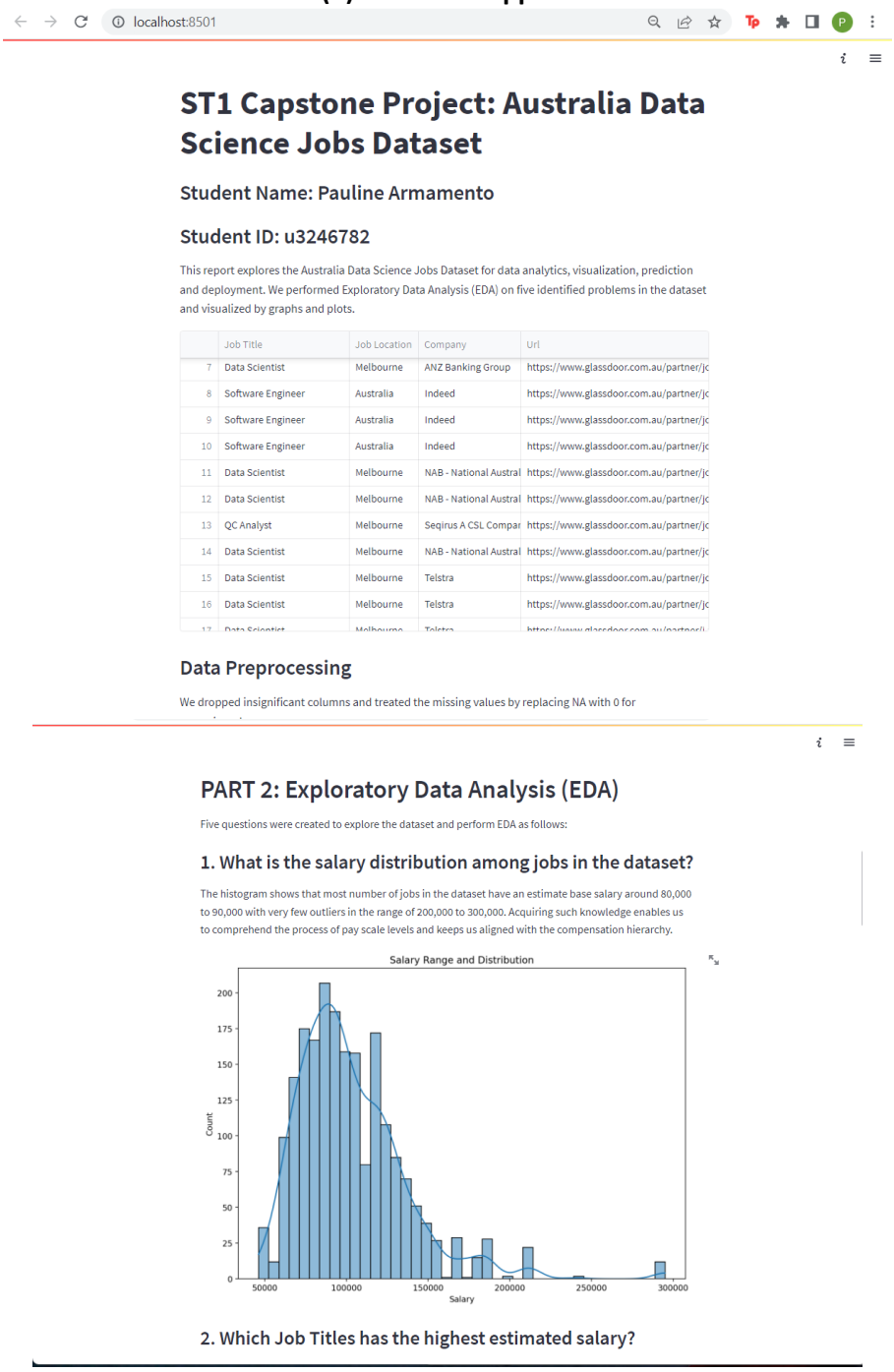
Figure 3
Job Roles by Count
a scientist
ta analyst
specialist
engineer
consultant
manager
0
25
50
75
100
125
150
175
200
Count
x=47.0 y=


Figure 4
Top High Paying State by Estimate Base Salary
Average Estimated Base Salary ($)
0
20000
40000
60000
80000
100000
Victoria
Tasmania
x= y=8.83e+04


Figure 5
Correlation Heatmap of Data Science Job Variables
Estimate Base Salary
Low Estimate
High Estimate
Company Rating
Company CEO Approval
Company Career Opportunities
Company Culture and Values
Company Work Life Balance
1.0
0.8
0.6
0.4
0.2
0.0


ST1Capstone_Armamento_u3246782
"C:\Users\pauar\Desktop\UC\Software Technology 8995\ST1Capstone\venv\Scripts\python.exe" "C:/Users/pauar/Desktop/UC/Soft
Top 10 Job Titles by Average Estimate Base Salary:
Technical Fellow, Job Seeker: $295,000.00
IT Systems Specialist: $240,000.00
Senior Product Manager: $209,000.00
Lead Analytics Consultant: $181,275.00
Analytics Consultant: $172,885.33
Senior Analytics Consultant: $171,556.00
Senior UX Designer: $170,000.00
Senior Contract Administrator: $169,706.00
Software Engineer: $166,255.82
Lead Security Consultant: $157,500.00
Data scientist jobs are highest in Sydney with 207 job postings.
Data analyst jobs are highest in Melbourne with 72 job postings.
Data engineer jobs are highest in Melbourne with 28 job postings.
Data consultant jobs are highest in Brisbane with 11 job postings.
Data specialist jobs are highest in Melbourne with 47 job postings.
Data manager jobs are highest in Perth with 5 job postings.
Average Estimate Base Salary for each state
New South Wales: $111,057.46
Western Australia: $109,703.21
Northern Territory: $107,837.75
South Australia: $103,525.85
Tasmania: $103,128.13
Queensland: $98,723.35
Victoria: $94,633.06

**(b) Streamlit Application**




**Perform Classification (Predictive Analysis)**

In the Predictive Analytics component, we have incorporated the variables Low Estimate and High Estimate with high positive correlation as X input features. Additionally, we have also considered Company Career Opportunities a variable exhibiting highest negative correlation to depict an opposite directional movement. By encompassing all these variables together, we can extract supplementary insights about both target variable and its predictors.

We used the popular machine learning algorithm Random Forest Classifier[2] for this task due to its reputation for generating highly precise outcomes, it is able to handle missing values and outliers well making it a reliable choice.

**Implementation and Deployment**

We have chosen to use a web app Streamlit for the deployment of this project as it is easy to use and promotes collaboration making it a popular tool for data scientists and developers. Screenshots from the previous section gives us how the program visually works under streamlit.

In addition, the GitHub repository: https://github.com/parmamento/st1capstone was created to fulfill the requirements for this project.

## 4. Reflection

Doing this project showcased our knowledge in applying the concepts of Data Analytics, Visualization, Prediction and Deployment that we learned from this unit. Furthermore, it provided us with more practice in using popular python libraries such as pandas, numpy, seaborn and scikit-learn for data manipulation, visualization and machine learning tasks. The final outcome involved utilizing Random Forest Classifier algorithm to perform predictive analytics on estimate base salary variable for gaining valuable insights.

Reference:

[1] https://www.kaggle.com/datasets/nadzmiagthomas/australia-data-science-jobs?resource=download

[2] Cutler, Adele & Cutler, David & Stevens, John. (2011). Random Forests. 10.1007/978-1-4419-9326-7_5.