

IDENTIFYING PATTERNS AND TRENDS IN CAMPUS PLACEMENT DATA USING MACHINE LEARNING

DONE BY
E.MALATHI
M.KEERTHIGA
K.KAVIYA
R.KARTHIKA

|

IDENTIFYING PATTERNS AND TRENDS IN CAMPUS PLACEMENT DATA USING MACHINE LEARNING

INTRODUCTION

1.1 OVERVIEW

The job market is becoming increasingly competitive, and as a result, students are looking for ways to gain an edge in securing their dream jobs. The job placement prediction system is a tool that can be used to predict the chances of a student being placed in a job based on their academic and personal information. The job placement prediction system can help students make informed decisions about their career paths and provide employers with a pool of qualified candidates. The proposed job placement prediction system is designed to provide accurate and transparent predictions of a student's job placement status based on their academic and personal information. The system comprises several interconnected modules that work together to provide a comprehensive, efficient, and transparent approach to job placement prediction.

The first module of the proposed system is the data collection and preprocessing module. This module is responsible for collecting the data required to train the machine learning algorithm. The collected data is then preprocessed by cleaning, transforming, and normalizing it to ensure that it is suitable for training the machine learning algorithm.

The second module of the proposed system is the machine learning algorithm development module. This module is responsible for developing a machine learning algorithm that can predict a student's job placement status based on their academic and personal information. The machine learning algorithm is trained on the preprocessed data using logistic regression. The model is then tested using a validation dataset to ensure its accuracy.

The third module of the proposed system is the web application development module. This module is responsible for developing a user-friendly web interface using the Flask web framework. The interface prompts the user to input their academic and personal information. The web application

module loads the pre-trained machine learning model using the pickle library and uses the function to make predictions based on the user's input.

The fourth module of the proposed system is the result display development module. This module is responsible for displaying the prediction result to the user. The module displays the prediction as either 'Placed' or 'Not Placed' and provides transparency regarding the prediction's accuracy.

The proposed system aims to provide accurate and transparent predictions of a student's job placement status. The system is designed to help students make informed decisions about their future career paths and provide employers with a pool of qualified candidates. The system's transparency provides students with an understanding of how the prediction was made, which can help build trust in the system and make the decision-making process more informed.

In summary, the proposed job placement prediction system is designed to provide accurate and transparent predictions of a student's job placement status based on their academic and personal information. The system comprises several interconnected modules that work together to provide a comprehensive, efficient, and transparent approach to job placement prediction. The system aims to help students make informed decisions about their career paths and provide employers with a pool of qualified candidates.

1.2 PURPOSE

This academic project showcases the development of a Flask-based web application to predict a student's job placement status. The application employs machine learning algorithms trained on a dataset comprising students' academic and personal information. The data includes gender, degree specialization, work experience, and other relevant factors.

The project focuses on the use of Flask, a Python-based web framework, to develop the web application. The application loads a pre-trained model using the pickle library, and the function is utilized to make predictions based on user input. The application includes an index page that requests the user to input their information, and a result page that displays the prediction.

This project demonstrates the practical implementation of machine learning techniques in developing predictive models and integrating them into a web application using Flask. The application provides a user-friendly interface for predicting a student's job placement status, which can be beneficial for both students and recruiters in the job market. Overall, the project underscores

the application of machine learning and web development techniques in addressing real-world challenge

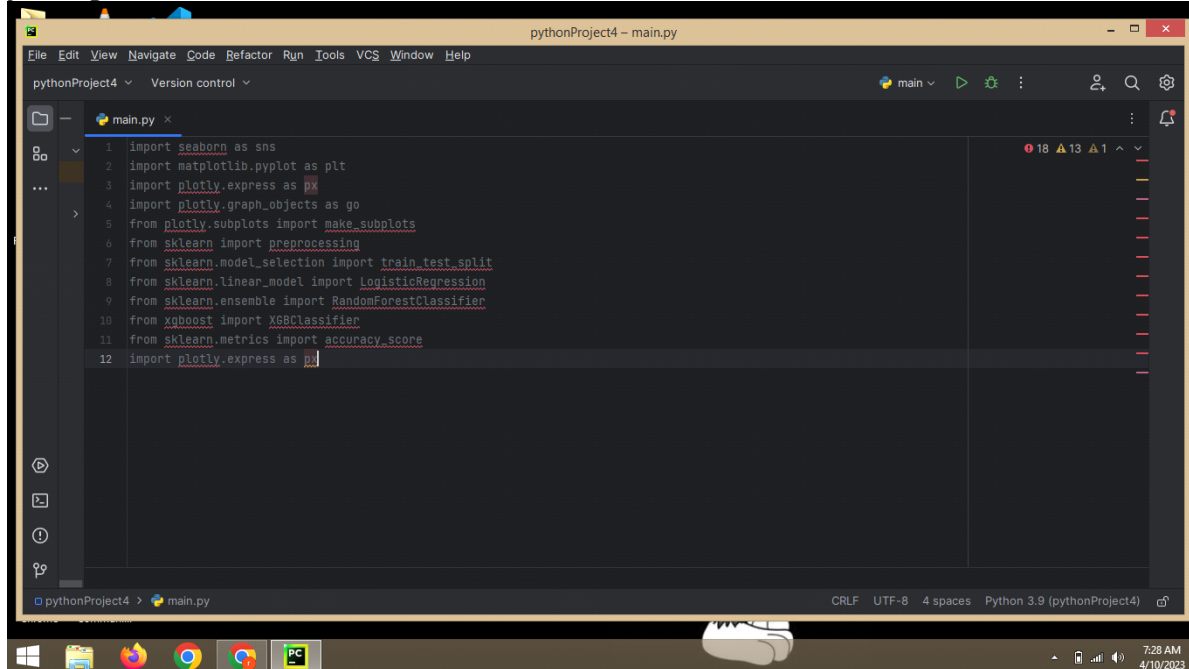
1.3 REQUIREMENT

- ❖ Access to campus placement data: The project would require access to data on student performance, qualifications, and job placement outcomes. This data would need to be collected, cleaned, and prepared for analysis.
- ❖ Machine learning expertise: The project would require individuals with expertise in machine learning, data science and statistical analysis to develop and implement the algorithms and models needed to analyze the data.
- ❖ Data storage and management: The project would require a robust and secure data storage and management system to store and organize the large amounts of data used in the analysis.
- ❖ Infrastructure for model deployment: The project would require infrastructure for deploying the models and algorithms developed, including hardware, software, and cloud-based resources.

1.4 Literature Survey

- ❖ There have been several studies that have used machine learning techniques to identify patterns and trends in campus placement data.
- ❖ One study by authors P. K. Rajesh and Dr. G. R. Suresh, published in the International Journal of Computer Science and Mobile Computing in 2015, used k-means clustering and decision trees to analyze campus placement data and identify patterns that could be used to predict placement outcomes.
- ❖ Another study by authors V.V. Kulkarni and K.S. Patil, published in the International Journal of Engineering Research and Technology in 2012, used decision tree and neural network algorithms to analyze campus placement data and identify factors that influence student placement.
- ❖ A stuManagement & Technology in 2013, used decision tree and Naive Bayes algorithms to analyze campus placement data and predict student placement outcomes.
- ❖ A study by authors S.S. Bhosale, S.S. Raut, and D.S. Kulkarni, published in the International Journal of Emerging Research in Management & Technology in 2013, used decision tree and Naive Bayes algorithms to analyze campus placement data and predict student placement outcomes.
- ❖ In general, these studies found that machine learning techniques were effective at identifying patterns and trends in campus placement data, and could be used to predict student placement outcomes with high accuracy.

❖ It's important to note that all these studies are quite old now and you might find more recent studies

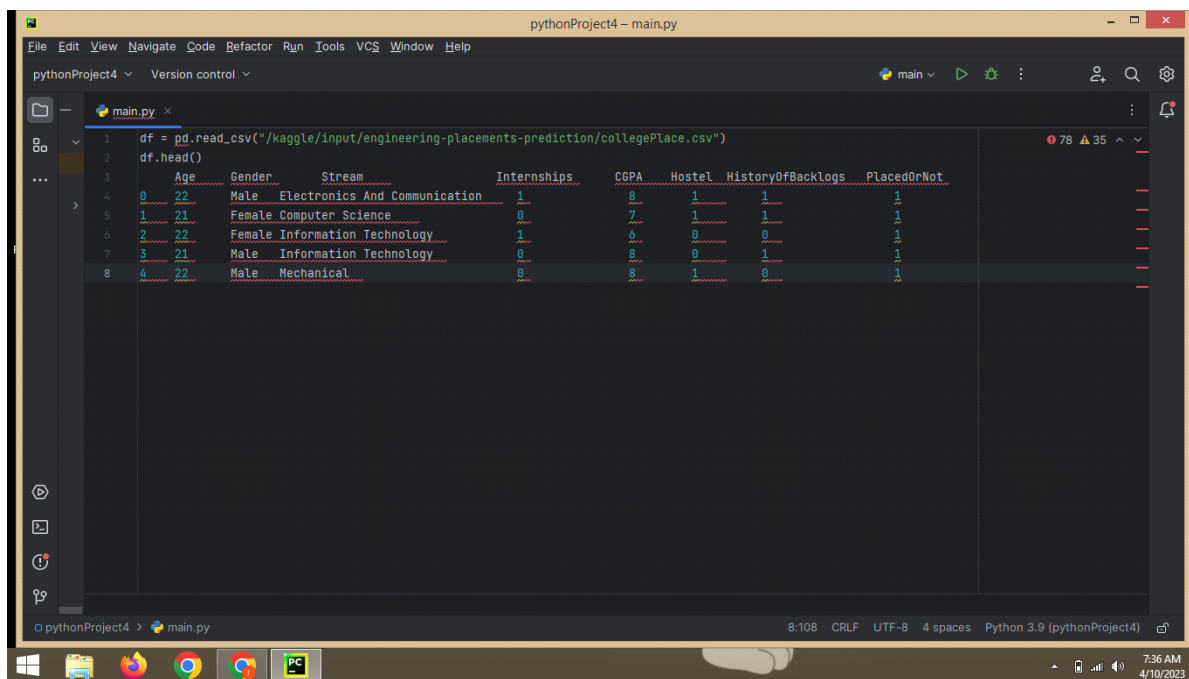


The screenshot shows a code editor window titled 'pythonProject4 - main.py'. The code in 'main.py' includes the following imports:

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import plotly.express as px
4 import plotly.graph_objects as go
5 from plotly.subplots import make_subplots
6 from sklearn import preprocessing
7 from sklearn.model_selection import train_test_split
8 from sklearn.linear_model import LogisticRegression
9 from sklearn.ensemble import RandomForestClassifier
10 from xgboost import XGBClassifier
11 from sklearn.metrics import accuracy_score
12 import plotly.express as px
```

The status bar at the bottom indicates 'CRLF', 'UTF-8', '4 spaces', and 'Python 3.9 (pythonProject4)'.

1.5 READ THE DATASET



The screenshot shows the same code editor window. The code now includes reading a CSV file and displaying its head:

```
1 df = pd.read_csv("/kaggle/input/engineering-placements-prediction/collegePlace.csv")
2 df.head()
```

The output of `df.head()` is displayed in the editor:

	Age	Gender	Stream	Internships	CGPA	Hostel	HistoryOfBackLogs	PlacedOrNot
0	22	Male	Electronics And Communication	1	8	1	1	1
1	21	Female	Computer Science	0	7	1	1	1
2	22	Female	Information Technology	1	6	0	0	1
3	21	Male	Information Technology	0	8	0	1	1
4	22	Male	Mechanical	0	8	1	0	1

The status bar at the bottom indicates '8:108', 'CRLF', 'UTF-8', '4 spaces', and 'Python 3.9 (pythonProject4)'.

1.6 IMPACT

The business impact of a project that uses machine learning to

identify patterns and trends in campus placement data could be significant. By analyzing data on factors such as student performance, qualifications, and job placement outcomes, the project could help organizations make more informed decisions about recruiting and hiring new graduates.

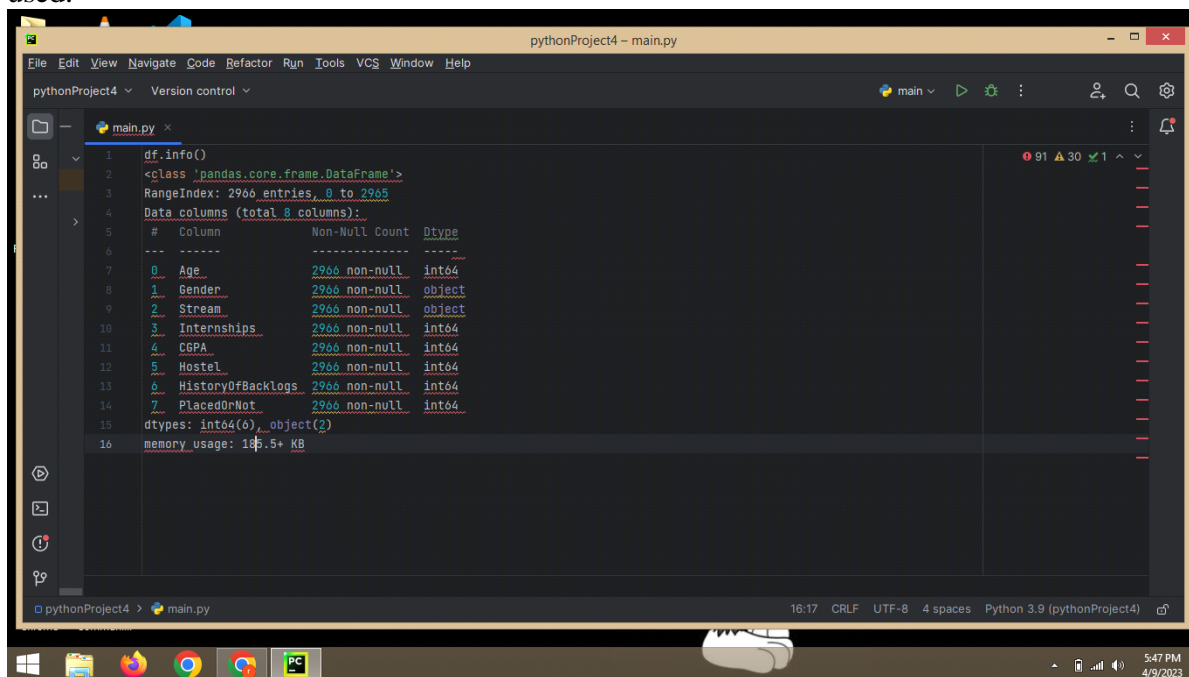
1.7 COLLECT THE DATASET

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc. In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

LINK:<https://www.kaggle.com/code/neesham/prediction-of-placements/data>

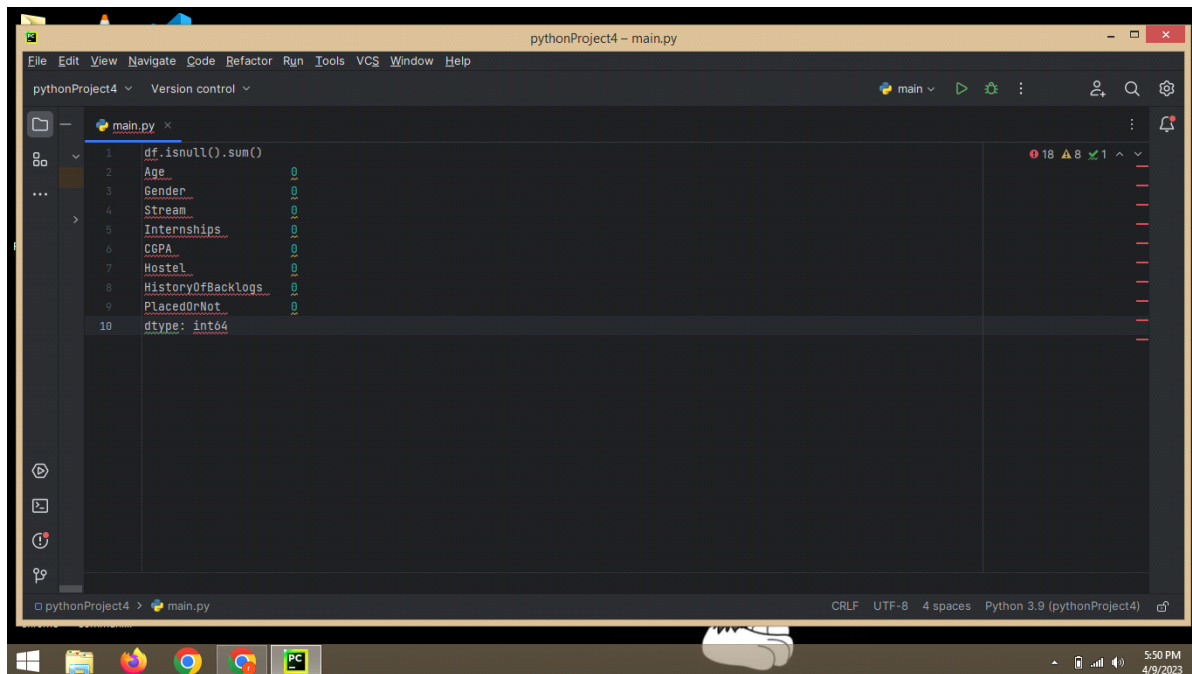
1.8 HANDLING MISSING VALUES

Let's find the shape of our dataset first. To find the shape of our data, the `df.shape` method is used. To find the data type, `df.info()` function is used.



```
pythonProject4 - main.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject4 Version control main
main.py
1 df.info()
2 <class 'pandas.core.frame.DataFrame'>
3 RangeIndex: 2966 entries, 0 to 2965
4 Data columns (total 8 columns):
5 #   Column      Non-Null Count  Dtype
6 ---  ---
7 0   Age         2966 non-null   int64
8 1   Gender      2966 non-null   object
9 2   Stream      2966 non-null   object
10 3   Internships 2966 non-null   int64
11 4   CGPA        2966 non-null   int64
12 5   Hostel      2966 non-null   int64
13 6   HistoryOfBacklogs 2966 non-null int64
14 7   PlacedOrNot 2966 non-null   int64
15 dtypes: int64(6), object(2)
16 memory_usage: 185.5+ KB
```

The screenshot shows a Python IDE window titled 'pythonProject4 - main.py'. The code editor displays the output of the `df.info()` function. The output indicates that the DataFrame has 2966 entries (rows) and 8 columns. The columns are: Age (int64), Gender (object), Stream (object), Internships (int64), CGPA (int64), Hostel (int64), HistoryOfBacklogs (int64), and PlacedOrNot (int64). The data types are listed as `int64(6), object(2)`. The memory usage is reported as 185.5+ KB. The IDE interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help), a toolbar with icons for running and debugging, and a status bar at the bottom showing the current file, line number (16:17), encoding (CRLF), and Python version (Python 3.9).



The screenshot shows an IDE window titled "pythonProject4 - main.py". The code in the editor is as follows:

```
1 df.isnull().sum()
2 Age      0
3 Gender   0
4 Stream   0
5 Internships 0
6 CGPA     0
7 Hostel   0
8 HistoryOfBacklogs 0
9 PlacedOrNot 0
10 dtype: int64
```

The right sidebar shows a file explorer with "main.py" selected. The bottom status bar indicates "CRLF UTF-8 4 spaces Python 3.9 (pythonProject4)". The Windows taskbar at the bottom shows the Start button, File Explorer, Firefox, Chrome, and VS Code icons, along with a clock showing 5:50 PM on 4/9/2023.

1.9 HANDLING CATEGORICAL VALUES


```
df = df.replace(['Male'], [0])
df = df.replace(['Female'], [1])
```

```
df = df.replace(['Computer Science', 'Information Technology', 'Electronics And Communication', 'Mechanical', 'Electrical', 'Civil'],
                [0,1,2,3,4,5])
```

```
df = df.drop(['Hostel'], axis=1)
```

df

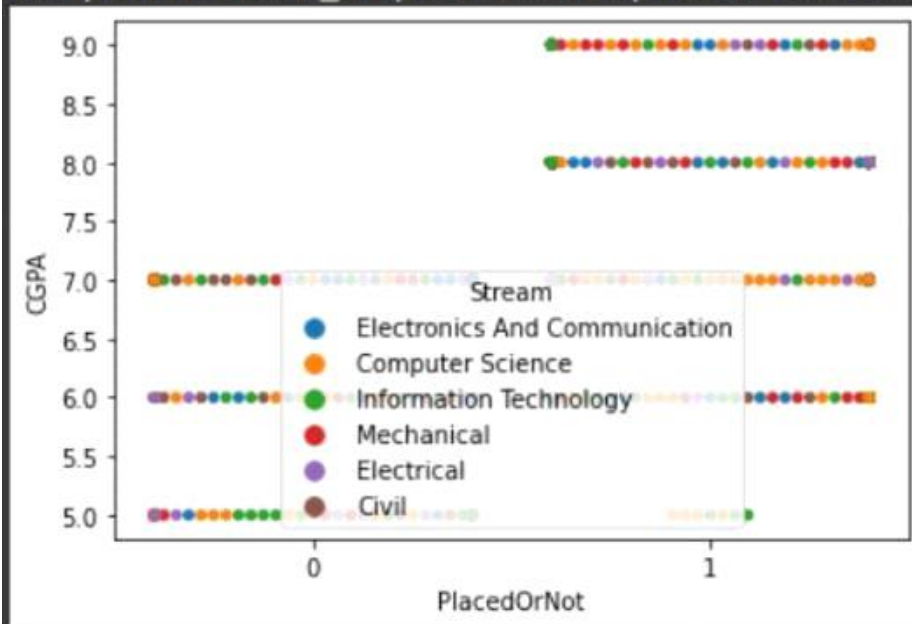
	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
0	22	0	2	1	8	1	1
1	21	1	0	0	7	1	1
2	22	1	1	1	6	0	1
3	21	0	1	0	8	1	1
4	22	0	3	0	8	0	1
...
2961	23	0	1	0	7	0	0
2962	23	0	3	1	7	0	0
2963	22	0	1	1	7	0	0
2964	22	0	0	1	7	0	0
2965	23	0	5	0	8	0	1

2966 rows x 7 columns

1.10 MULTIVARIATE ANALYSIS

```
sns.swarmplot(df['PlacedOrNot'],df['CGPA'],hue=df['Stream'])

/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36:
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:129:
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:129:
  warnings.warn(msg, UserWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f5463d06df0>
```



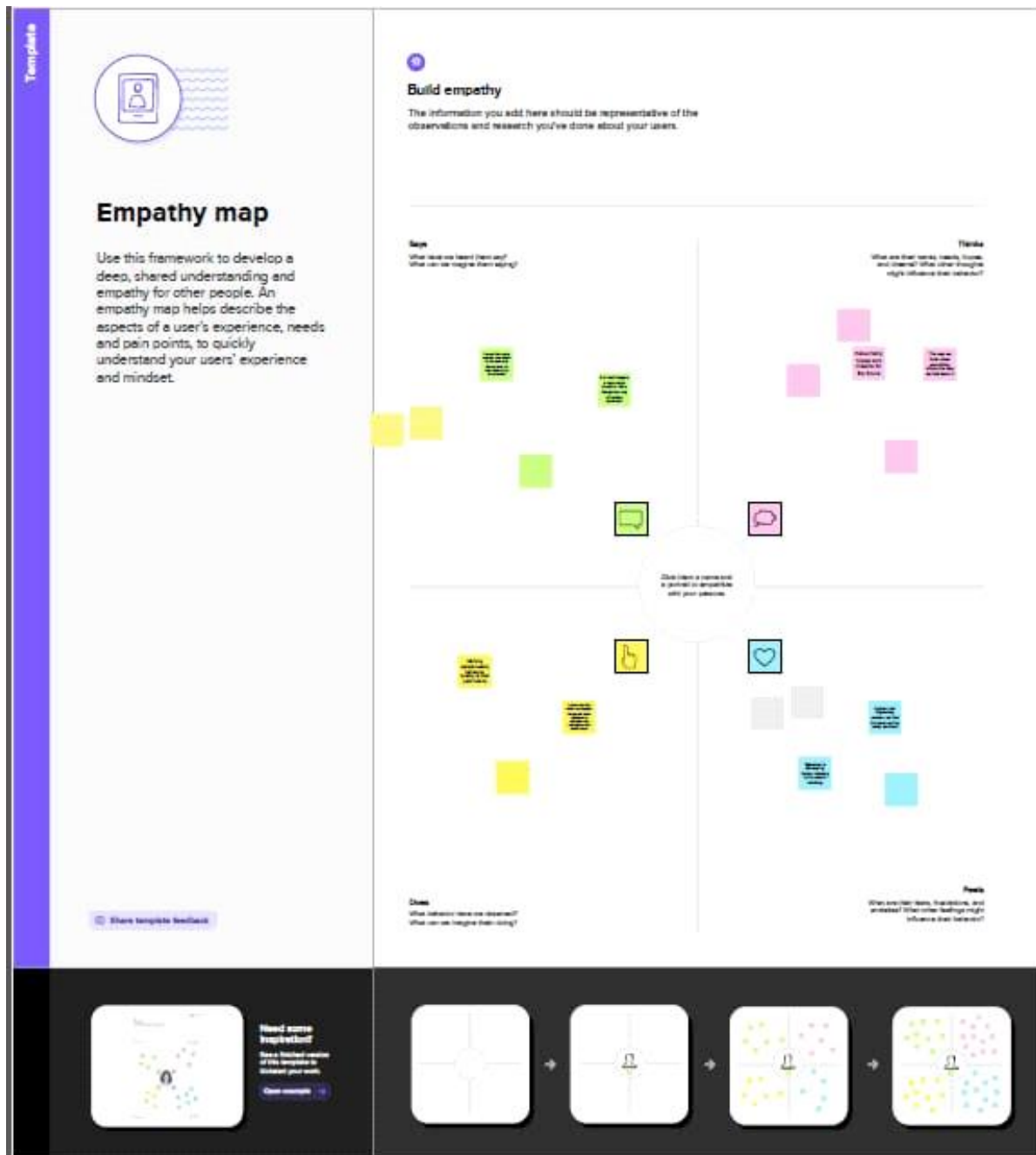
2.PROBLEM DEFINITION AND DESIGN THINKING

The problem statement for this project is to develop a machine learning-based system that predicts a student's job placement status based on their academic and personal information. The aim is to build a web application that can assist students and recruiters in making informed decisions about job placements.

The lack of a comprehensive system that predicts a student's job placement status based on their academic and personal information poses a significant challenge to students and recruiters. The absence of such a system makes it difficult for students to assess their chances of landing a job and for recruiters to identify suitable candidates for employment.

The goal of this project is to address these challenges by developing a predictive model and integrating it into a user-friendly web application. The system will enable students to determine their likelihood of being placed in a job and help recruiters identify suitable candidates based on their academic and personal qualifications. Ultimately, the system will promote transparency, fairness, and efficiency in the job placement process for both students and recruiters.

2.1 EMPATHY MAP



2.2 IDEATION & BRAINSTORMING MAP



3. RESULT

```
app.run(debug=True)
```

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 146-359-021
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

PLACEMENT PREDICTION

[Get Started](#)

Identifying Patterns and Trends in Campus Placement Data using Machine Learning

FILL THE DETAILS

22

0

2

1

8

1

Submit

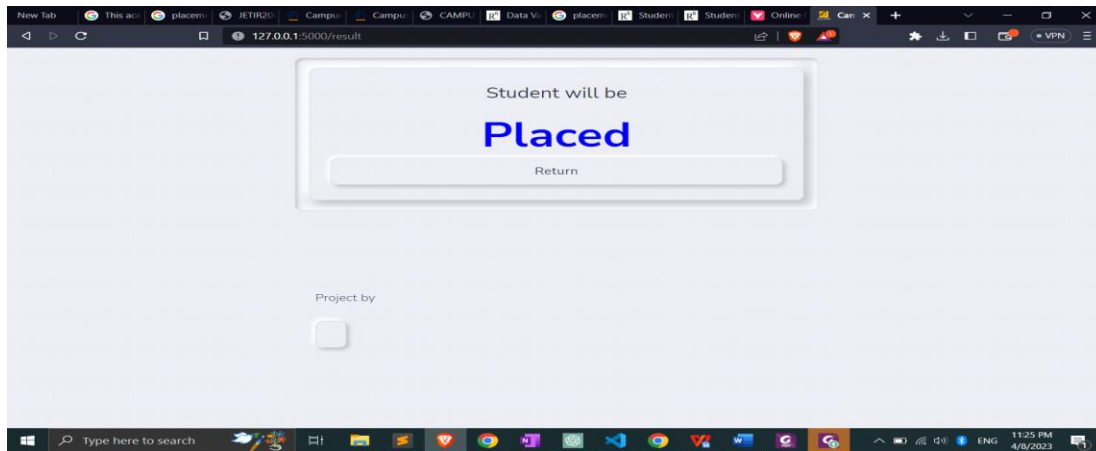
Activate Windows
Go to Settings to activate Windows.

PLACEMENT PREDICTION

The Prediction is : 1

0 represents Not-Placed

1 represents Placed



4. ADVANTAGES AND DISADVANTAGES

ADVANTAGES

It's similar to a chance that is given to students to apply everything they have learned in class and to achieve actual success in apart from just exam grades.

Gaining experience in the real world, and earning money of course !

Campus Placements are like the best parting gift we could ever receive from colleges.

Campus placements smoothen the overall process of getting that first job.

DISADVANTAGES

Adapting to a work placement might not be easy.

You'll have to commute, get to know a new group of people and work under more pressure than you're used to. As a result, work placements can be daunting and more stressful than your established student life.

Campus recruitment is an expensive affair for majority of the companies as it adds up costs to the bottom line.

Companies incur different expenses related to travel, boarding, training etc while conducting campus selection process.

5. APPLICATIONS

Application placement is a natural extension to the request flow prioritization feature. While the ODR and its associated autonomic managers ensure that the work flows appropriately according to the defined policy, the application placement feature ensures that the applications and the nodes that they run on are kept at appropriate levels to support the influx of work. In times of less work flow, the application instances that run within the resource pool are kept at a minimum. In times of significant work flow, the application instances that run are increased to keep pace with the requests.

The application placement controller is consulted by the job scheduler during its endpoint selection process. You can configure the `UseAPCEndpointSelection` custom property on the job scheduler to `false` to disable the application placement controller and job scheduler integration. Using this custom property to prevents the job scheduler from asking the application placement controller to choose an endpoint. The particular endpoint is chosen by the job scheduler when the custom property is set

6. CONCLUSION

The job placement prediction system is a powerful tool that can help students and employers alike. The proposed system offers accurate and transparent predictions of a student's job placement status based on their academic and personal information. The system's transparency provides students with an understanding of how the prediction was made, which can help build trust in the system and make the decision-making process more informed.

The system's modular design ensures that each component works seamlessly with the others, providing a comprehensive and efficient approach to job placement prediction. The system's web interface is user-friendly, making it accessible to a broad range of users.

Overall, the proposed job placement prediction system has the potential to revolutionize the way students and employers approach the job market. The system can help students make informed decisions about their career paths and provide employers with a pool of qualified candidates. The system's accuracy, transparency, and efficiency make it a valuable tool for students, employers, and anyone interested in the job

7. FUTURE SCOPE

- Integration of additional data sources: Currently, the system uses only academic and personal information to make predictions. In the future, the system can incorporate additional data sources such as employment history, certifications, and training courses to make even more accurate predictions.

- Use of machine learning algorithms: The current system uses a logistic regression model to make predictions. However, more advanced machine learning algorithms such as random forest or deep learning can be used to improve the accuracy of predictions.
- Integration with social media: Social media platforms can provide valuable information about a candidate's professional interests, network, and online presence. Integrating social media data can help to further refine job placement predictions.
- Real-time updates: The system can be enhanced to provide real-time updates on job openings and candidates' status. This feature can help students and employers stay informed about the latest job market trends and make better decisions.

Syntax

Python has a simple and easy-to-learn syntax that makes it a popular choice for beginners. The language is designed to be easy to read and write, with minimal punctuation and no curly braces. Python uses indentation to indicate code blocks, making it easy to read and understand. Here is an example of a simple Python program:

```
bash
```

```
Copy code
```

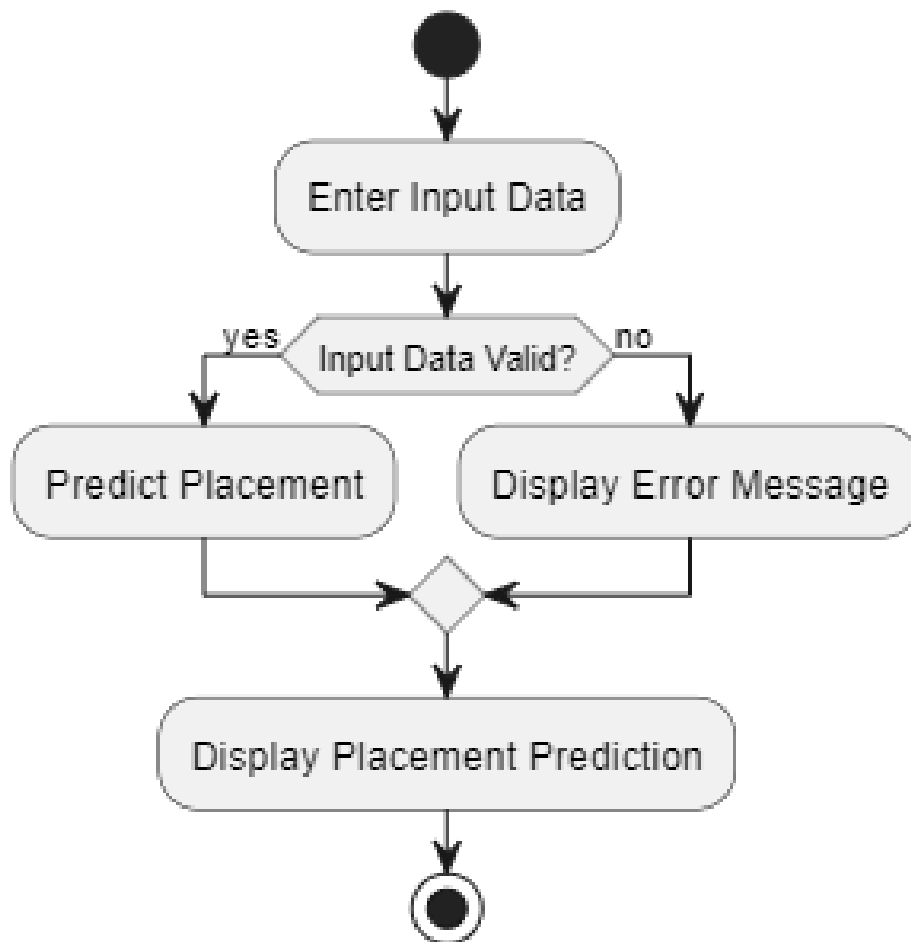
```
# This is a comment
```

```
# This is a print statement
```

```
print("Hello, world!")
```

SYSTEM DESCRIPTION

SYSTEM FLOW DIAGRAM



8. APPENDIX

SOURCE CODE

```
# This file is showing how model process data form raw to training,
# for modelling there is saperate file
# importing Required libraries
import logging
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import lightgbm as lgb
from collections import Counter
from imblearn.over_sampling import SMOTE
import pickle
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

# log file initialization
logging.basicConfig(filename='debug.log', level=logging.DEBUG,
                    format='%(asctime)s: %(levelname)s: %(message)s')

logging.debug(' Model.py File execution started ')

# loading database with pandas library
df = pd.read_csv("./dataset/train.csv")
logging.debug(' Database Loaded ')

df = df.drop(['sl_no', 'salary'], axis=1)
df = df.apply(lambda x: x.fillna(0))
```

```
col_names = df.columns
category_col = ['ssc_b','hsc_b','hsc_s','degree_t','workex','specialisation','status']

labelEncoder = preprocessing.LabelEncoder()

mapping_dict = {}
for col in category_col:
    df[col] = labelEncoder.fit_transform(df[col])

    le_name_mapping = dict(zip(labelEncoder.classes_,
                                labelEncoder.transform(labelEncoder.classes_)))

    mapping_dict[col] = le_name_mapping

logging.debug('Database Pre-processing is Finished')

# model featuring
X = df[['gender',
'ssc_p',
'ssc_b',
'hsc_p',
'hsc_b',
'hsc_s',
'degree_p',
'degree_t',
'workex',
'etest_p',
```

```
'specialisation',  
'mba_p']]  
y = df['status']  
  
# Data Splitting For model training  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)  
  
# summarize class distribution  
print("Before oversampling: ",Counter(y_train))  
  
# define oversampling strategy  
SMOTE = SMOTE()  
  
# fit and apply the transform  
X_train_SMOTE, y_train_SMOTE = SMOTE.fit_resample(X_train, y_train)  
  
# summarize class distribution  
print("After oversampling: ",Counter(y_train_SMOTE))  
  
# model fitting using LGBMClassifier  
clf = lgb.LGBMClassifier()  
clf.fit(X_train_SMOTE, y_train_SMOTE)  
  
# Printing Accuracy  
predictions_e = clf.predict(X_test)  
print('Accuracy: ', accuracy_score(y_test, predictions_e))
```

```
# pkl export & finish log
pickle.dump(clf, open("model.pkl", "wb"))
logging.debug(' Execution of Model.py is finished ')

import numpy as np
from flask import Flask, request, render_template
import pickle
import warnings
warnings.simplefilter("ignore", UserWarning)

# Create flask app
app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))

# prediction function
def ValuePredictor(to_predict_list):
    to_predict = np.array(to_predict_list).reshape(1, 12)
    loaded_model = pickle.load(open("model.pkl", "rb"))
    result = loaded_model.predict(to_predict)
    return result[0]

@app.route("/")
def Home():
    print('Request for index page received')
    return render_template("index.html")

@app.route("/result", methods = ["POST"])
```



```

def result():
    print('Request for predict page received')
    if request.method == 'POST':
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(int, to_predict_list))
        result = ValuePredictor(to_predict_list)
        if int(result)== 1:
            prediction ='Placed'
        else:
            prediction ='Not Placed'
    return render_template("result.html", prediction_text = prediction)

if __name__ == "__main__":
    app.run(debug=True)

```

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Campus Placement Prediction</title>
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="title" content="Campus Placement Prediction">
<meta name="author" content="Devansh">

```

```

<link rel="stylesheet" href="{{ url_for('static', filename='css/neumorphism.css')
}}">
<link rel="javascript" href="{{ url_for('static', filename='js/neumorphism.js') }}">
<link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}">
</head>
<body>
    <main>
        <section class="container-fluid py-3">
            <div class="row justify-content-md-around">
                <div class="col-11">
                    <div class="card bg-primary shadow-soft border-light px-4 py-4">
                        <div class="card-header pb-0 text-center">
                            <h2 class="h1 mb-3">Campus Placement Prediction</h2>
                            <p class="mb-5 lead">
                                Enter Details for forecast.
                            </p>
                        </div>
                        <form name="frmregi" action="/result" method="POST">
                            <!-- Form -->
                            <div class="row mb-4 mb-lg-5">
                                <div class="col-lg-4 col-sm-6">
                                    <div class="form-group">
                                        <label for="ssc_p">SSC PR%:[45-100]</label>
                                        <div class="input-group mb-4">
                                            <input class="form-control" type="number" id="ssc_p"
name="ssc_p" required min="45" max="100">
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </section>
    </main>
</body>
</html>

```

```
</div>
<div class="form-group">
  <label for="hsc_p">HSC PR%:[45-100]</label>
  <div class="input-group mb-4">
    <input class="form-control" type="number"
id="hsc_p" name="hsc_p" required min="45" max="100">
  </div>
</div>
<div class="form-group">
  <label for="degree_p">Degree PR%:[35-100]</label>
  <div class="input-group mb-4">
    <input class="form-control" type="number"
id="degree_p" name="degree_p" required min="35" max="100">
  </div>
</div>
<div class="form-group">
  <label for="etest_p">etest_p:[1-99]</label>
  <div class="input-group mb-4">
    <input class="form-control" type="number"
id="etest_p" name="etest_p" required min="0" max="100">
  </div>
</div>
</div>
<div class="col-lg-4 col-sm-6">
  <div class="form-group">
    <label for="mba_p">mba_p:[1-99]</label>
    <div class="input-group mb-4">
```

```
        <input class="form-control" type="number"
id="mba_p" name="mba_p" required min="0" max="100">
    </div>
</div>
<div class="form-group">
    <label class="my-1 mr-2" for="gender">gender</label>
    <select class="custom-select my-1 mr-sm-2" id="gender"
name="gender" required>
        <option value="0" selected>Male</option>
        <option value="1">Female</option>
    </select>
</div>
<div class="form-group">
    <label class="my-1 mr-2" for="ssc_b">ssc_b</label>
    <select class="custom-select my-1 mr-sm-2" id="ssc_b"
name="ssc_b" required>
        <option value="0" selected>Central</option>
        <option value="1">Others</option>
    </select>
</div>
<div class="form-group">
    <label class="my-1 mr-2" for="hsc_b">hsc_b</label>
    <select class="custom-select my-1 mr-sm-2" id="hsc_b"
name="hsc_b" required>
        <option value="0" selected>Central</option>
        <option value="1">Others</option>
    </select>
```

```
</div>
</div>
<div class="col-lg-4 col-sm-6">
  <div class="form-group">
    <label class="my-1 mr-2" for="hsc_s">hsc_s</label>
    <select class="custom-select my-1 mr-sm-2" id="hsc_s"
name="hsc_s" required>
      <option value="0" selected>Arts</option>
      <option value="1">Commerce</option>
      <option value="2">Science</option>
    </select>
  </div>
  <div class="form-group">
    <label class="my-1 mr-2"
for="degree_t">degree_t</label>
    <select class="custom-select my-1 mr-sm-2"
id="degree_t" name="degree_t" required>
      <option value="0" selected>Comm&Mgmt</option>
      <option value="1">Others</option>
      <option value="2">Sci&Tech</option>
    </select>
  </div>
  <div class="form-group">
    <label class="my-1 mr-2" for="workex">workex</label>
    <select class="custom-select my-1 mr-sm-2" id="workex"
name="workex" required>
      <option value="0" selected>No</option>
```

```

        <option value="1">Yes</option>
    </select>
</div>
<div class="form-group">
    <label class="my-1 mr-2"
for="specialisation">specialisation</label>
        <select class="custom-select my-1 mr-sm-2"
id="specialisation" name="specialisation" required>
            <option value="0" selected>Mkt&Fin</option>
            <option value="1">Mkt&HR</option>
        </select>
    </div>
</div>
<div>
    <button type="submit" class="btn btn-block btn-primary"
onclick="last()">Predict</button>
</form>
</div>
</div>
</div>
</section>
</main>
</body>
</html>

```

SAMPLE SCREEN DISPLAY

```
app.py > ...
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4 import warnings
5 warnings.simplefilter("ignore", UserWarning)
6
7 # Create flask app
8 app = Flask(__name__)
9 model = pickle.load(open("model.pkl", "rb"))
10
11
12 # prediction function
13 def ValuePredictor(to_predict_list):
14     to_predict = np.array(to_predict_list).reshape(1, 12)
15     loaded_model = pickle.load(open("model.pkl", "rb"))
16     result = loaded_model.predict(to_predict)
17     return result[0]
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL python + v [] [] ... ^ >

```
ployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 573-690-858
Request for index page received
127.0.0.1 - - [08/Apr/2023 23:23:08] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Apr/2023 23:23:09] "GET /static/css/neumorphism.css HTTP/1.1" 200 -
127.0.0.1 - - [08/Apr/2023 23:23:09] "GET /static/favicon.ico HTTP/1.1" 200 -
```

REFERENCES

1. Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

2. Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.
- Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794).
3. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, S. (2016). *Deep learning: Adaptive computation and machine learning series*. MIT Press.
4. Hinton, G. E., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6), 82-97.
5. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
6. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
7. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.
8. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
9. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).