School of Science and Technology

COMP40721

FOUNDATION OF ARTIFICIAL INTELLIGENCE

Coursework Title: Implementation and Written Report

Project Title: Machine Learning Techniques to analyse the "Stack Overflow 2023 Annual Developer Survey".

Course Leader

Prof. Salisu Yahaya

Submitted by

Bhagyesh Ravindra Chaudhari

NTU ID: N1218683

‘

1. Introduction

The primary goal of this assessment in this coursework is to investigate developer compensation using a dataset that differentiates between high and low compensation levels. If your annual salary is less than $50000, you fall into this category. As the dataset provides, conversion to USD is required for compatible analysis. ConvertedCompYearly values in this dataset are used to discover what influences salary differences among developers in different locations, as well as trends that affect these differences.

CRISP-DM, or Cross Industry Standard Process Data Mining, is a well-known framework that is widely used in data mining and analytics. It provides a six-phase structure for handling complex data analysis tasks in an organised manner. In this course, we used CRISP-DM to analyse and interpret the data collected by the stack overflow 2023 annual developer survey. It accelerates data preparation, analysis, and exploration by simplifying structured tasks. This task entails exploratory analysis, cluster analysis, and compensation predictions based on classification.

The application of all phases of CRISP-DM in the coursework is explained below:

1.1 Business Understanding: This coursework specification contained an explanation for the targets during this stage that required predicting the income spectrum for developers and analysing every factor which might influence competition. The coursework requirement specified the compensation is to separate into two categories. So, new column called “CompensationSalary” has been created to allow to achieve it. As below:

```
[ ]:
# Creating a new Categorical feature 'Compensation Salary' based on 'Salary' column
survey_data['CompensationSalary'] = survey_data['Salary'].apply(lambda x: 'Low' if x < 50000 else 'High')
```

1.2 Data Understanding: to visually illustrate and understand all aspects during comprehensive manner, Exploratory Data Analysis has been used during this coursework step by step. For appropriate visualization.

```
# Display the first few records of the dataset
survey_data.head()
```

```
survey_data.describe()
```

```
survey_data.dtypes
```

1.3 Data Preparation: In this a process of obtaining raw data available for further processing and analysis is referred to data preparation. Using this, I’m able to do all the necessary task to create the final dataset from the initial data source are covered within the data preparation step. In this step I used a function to map ‘Age’, ‘YearsCodePro’ and ‘EdLevel’ Categorical to Numerical values.

1.4 Modelling: The foundation of data analysis is data modelling. A model generates the desired outcome through the input from organised data. Selecting the appropriate model type is the task of this stage, irrespective of either the challenge is one of clustering, regression, or classification. After picking the model group using the set of algorithms within that selected group.

1.5 Model Evaluation: In this Step, the model is evaluated by to determine if the model is prepared for deployment. The model is evaluated using a collection of unexpected data and evaluated using a carefully developed set of evaluation criteria. Also need to confirm that the model correspondent to reality. In this case

I'm using the KNN Model, Decision Tree, and Logistic regression, whereas, training the data and testing the data have been used for ensemble learning.

1.6 Model Deployment: During a thorough assessment, the model eventually used in the desired structure and route. In the process of data, it is the last step. The process of data stated previously, takes careful thought during each step. For deploying this report is generated as applying model is not a part of this coursework.

2. Data Understanding, Data Preprocessing, Exploratory Data Analysis.

The knowledge is most likely the result of an international questionnaire or data collection process that involved developers to learn about their yearly salary in various currencies. The aim can be to understand worldwide trends of developer's salaries, while taking the variations in compensation due to aspect like experience, talents, and geography. This data could be useful for exploring differences in income and factors affecting pay in the developers' groups in various geographic locations and economic circumstances.

In this dataset their numerous columns. So, I'm selecting some appropriate features from the columns for analysis. Such as "Age, Employment, RemoteWork, EdLevel, YearsCodePro, WorkExp, DevType, Country, ProfessionalTech, Salary".

```
# Selecting the relevent features
survey_data = survey[['Age' ,'Employment','RemoteWork','EdLevel', 'YearsCodePro', 'WorkExp','DevType','Country','ProfessionalTech', 'Salary']]
```

1. **Age**: Identifies the connection that could occur between experience and age that might impact on Salary.
2. **Employment And RemoteWork**: determines the employer status about their jobs and remote work situations.
3. **EdLevel**: identifies the how education impact on carrier paths and income expectations.
4. **YearsCodePro & WorkExp**: This line highlights the relation between experience and more compensation. Evaluate the correlation between salary variance and several developer positions.
5. **DevType**: Investigates how developers position impact on the developer salary.
6. **Country**: explores the way differences in religion local economy affects compensation.
7. **ProfessionalTech**: evaluate the effects of professional ability on pay.
8. **ConvertedCompYearly:** The target variable for the analysis is salary. The primary goal is to comprehend the relationship between these characteristics and pay. The following snippet of the code shows that column 'Salary' is been renamed to CovertedCompYearly.

```
survey = survey.rename({'ConvertedCompYearly' : 'Salary'}, axis = 1)
```

Every feature that had been taken was done. so, after careful review of its recognised impact on developers group compensation. These features have been found to be important in determining salary. Differences, showing the complicated structure of variable impacting developers' salary across the world.

2.1 Displaying the NaN values: Raw dataset contains innumerable null values or cells which doesn't contain any values. These values mislead the visualization, as a result the plots can't provide the correct information. Following figure gives shows the snippet of the code and the output which shows number of missing values or null values in complete dataset. The next thing we did was to remove the null values from the compensation column.

```
survey_data.isna().sum()

Age                   0
Employment           12
RemoteWork           79
EdLevel               0
YearsCodePro        194
WorkExp           15381
DevType             115
Country               0
ProfessionalTech  16293
Salary                0
dtype: int64
```

```
survey = survey[survey['Salary'].notnull()]
```

2.2 Handling missing values (Data Cleaning): Handling the missing values is the important part of the data preparation pipeline. The raw data has numerous duplicate and NaN values. Using the dropna () function.

```
survey_data = survey_data.dropna(thresh=3)
survey_data.isna().sum()

print(survey_data.shape)

(47989, 11)
```

As result of this most of the null values from other columns are also removed. by mode functions. Such as, I replaced the categorical values to numerical values.

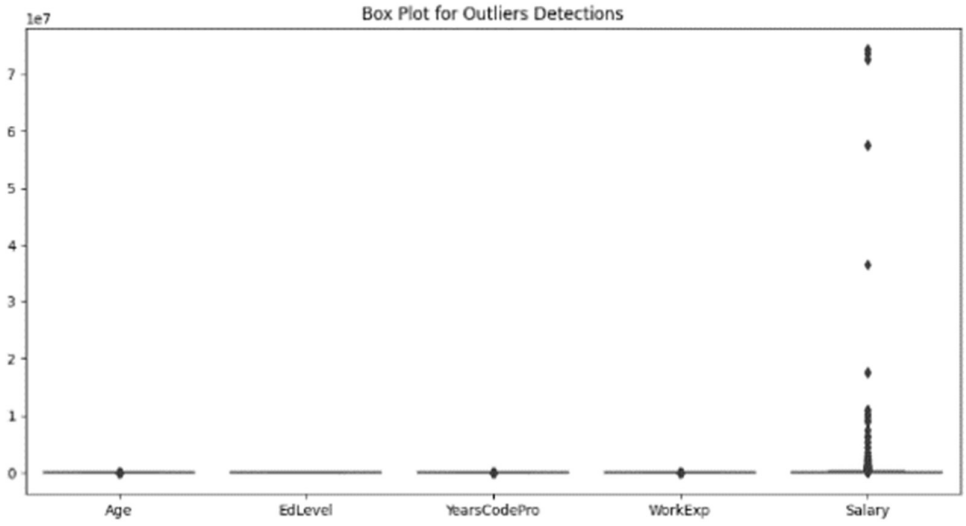
```
[ ]
survey_data['Employment'].fillna(survey_data['Employment'].mode()[0], inplace = True)
survey_data['EdLevel'].fillna(survey_data['EdLevel'].mode()[0], inplace = True)
survey_data['RemoteWork'].fillna(survey_data['RemoteWork'].mode()[0], inplace = True)
survey_data['YearsCodePro'].fillna(survey_data['YearsCodePro'].mode()[0], inplace = True)
survey_data['DevType'].fillna(survey_data['DevType'].mode()[0], inplace = True)
survey_data['Country'].fillna(survey_data['Country'].mode()[0], inplace = True)
survey_data['ProfessionalTech'].fillna(survey_data['ProfessionalTech'].mode()[0], inplace = True)
survey_data
```

Dividing the salary into two different categories: as per the information of the coursework, they stated clearly to salary need to be predicted in term of two categories.

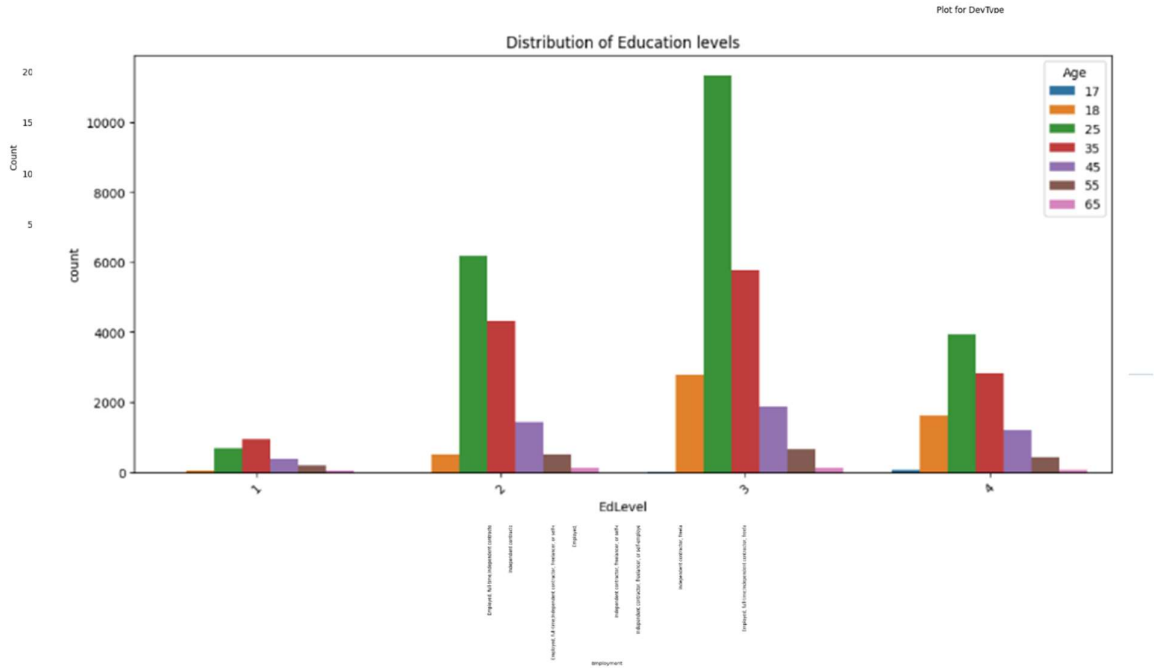
```
# Creating a new Categorical feature 'Compensation Salary' based on 'Salary' column

survey_data['CompensationSalary'] = survey_data['Salary'].apply(lambda x: 'Low' if x < 50000 else 'High')
```

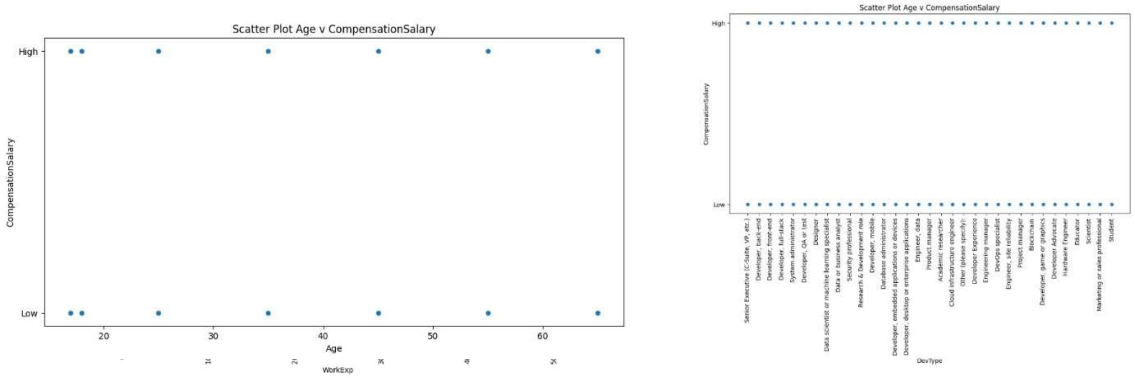
2.3 Identifying the outliers: Outliers must be identified as they can have a substantial influence on the integrity of results and skew the overall interpretation of data. The following figure depicts outliers in the given dataset.



2.4 Exploratory Data Analysis (EDA): It is and crucial step to understand the dataset and their relation between the features. With the help of EDA, we can identify the patterns and outliers. Histogram Plotting for the distribution of 'Age', 'Employment', 'WorkExp', and 'DevType':



2.5 Exploring the dataset through the scatter plot:



Above plot is generated regard of to find the relationship between the target variable. The plot represents the monitoring the dataset allow to visual inspection of potential patterns or outliers. I've generated the two-scatter plot to examine the relation between 'Age' with 'Salary' and 'DevType' with 'Salary'.

3. Cluster Analysis

Cluster Analysis is known for clustering as well. It is a method to use in data mining. The purpose of this method to used is in this coursework is to separate a dataset into categories such that the data point within each category are more like each other than to points in other categories. This operation uses for the Exploratory Data Analysis that can help us to identify the patterns and relationship with the data. There are so many algorithms like cluster analysis, k-means, hierarchical clustering, and density-based clustering. I've used the K-means Clustering to identify the unusual patterns or outliers in data and Hierarchical Clustering to visual representation of cluster relationships through dendrograms.

3.1 Data Transformation: In the process of transforming, cleaning, and arranging data into a format which can be used in evaluation to assist decision-making processes to accelerate grow a company is referred to data

transformation. With the help of Handling missing values and label Encoding I'm performing the Data Transformation. In this case the Data is categorical and after using label encoding the categorical data converted into the format suitable for Machine learning.

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()


obj_col = ['Employment','RemoteWork', 'DevType','WorkExp', 'Country', 'ProfessionalTech','CompensationSalary']
survey_data[obj_col]= survey_data[obj_col].apply(label_encoder.fit_transform)

survey_data
```

3.2 Normalization: It indicates the method used to produce something more predictable or common. mainly, it means the method during which notation and behaviour that can break the norms of society are accepted as "normal' in the field of Normalization. Through this I've applied the "StandardScaler" feature to train the Linear Regression and K-Nearest Neighbours model. in this feature have similar scales that prevent certain feature from influence the model between the model training process due to differences in scale.

```
# Normalisation
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

3.3 Performing the Cluster analysis: As I mentioned above, I'm implementing the K-means and Hierarchical Clustering for this dataset. Using K-Means and Hierarchical Clustering I use both distinct clustering methods, k-Means, and hierarchical clustering, to perform cluster analysis and provide insights into the fundamental framework of the dataset. The k-Means and Agglomerative Clustering modules from scikit-learn were applied to implement the analysis in the python.

3.3.1 K-Means Clustering: I've set the number of clusters (n_clusters) to 2 for k-Means clustering. suggesting that our objective the aim was to separate the data into two categories. Until integration, the approach maintained the cluster centroid by allocating the data points frequently to determine the nearest one. The scikit-learn implementations default options handle responsibility of the initiation method and terminating factor.

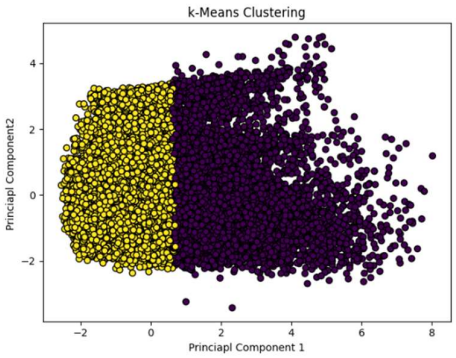


Figure 3.1

3.3.2 Hierarchical Clustering: In this, I've set the number of (n_cluster) to 2 in Hierarchical Clustering. The Agglomerative Clustering module by default implements the Ward linkage factors, and Hierarchical Clustering generates a tree-like structure of clusters. Clusters are continuously combined according to how close they became to get the require number of clusters using the hierarchical clustering method.

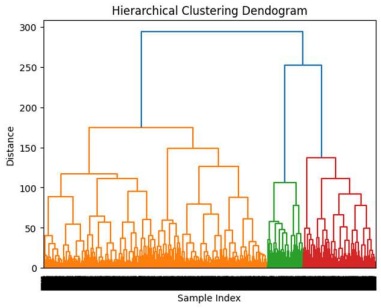


Figure 3.2

3.4 Optimal Number of Clusters: The most important aspect of cluster analysis is to identify the perfect number of clusters. Based on the features of the data and the fundamental framework I intend to show, so I decided to set the number of clusters two in our implementation. Cluster quality can be evaluated through the silhouette score.

3.5 Characteristics of Clusters:

3.5.1 K-Means Clustering: Using a fixed number of clusters of two, the technique is applied to k-Means clustering. This assumption is based on the potentiality that there are two distinct groups in the dataset. Both the training and test sets receive a cluster label once the model has been trained on the training set (X_train). The k-Means approach iteratively improves the clusters by reducing the sum of squares inside each cluster. By assigning each data point to the cluster with the closest centroid, it generates unique clusters. The specifics of each cluster are examined by computing the mean values of each characteristic inside each cluster as part of a post-processing phase. This analysis provides a comprehensive statistical overview of the feature distribution among the identified clusters, which is included in the 'kmeans_cluster_means' DataFrame.

3.5.2 Hierarchical Clustering: As I select two clusters for Hierarchical Clustering in this instance as well. Every data point is given a label based on the tree hierarchy that is established during the hierarchical clustering process. While k-Means separates the data into non-overlapping groups, the tree structure utilised in hierarchical clustering offers a more intricate representation of the relationships between the data. Using a similar post-processing method, we make clear the characteristics of every cluster that hierarchical clustering produces. By calculating the mean values of each characteristic inside each cluster, the main trends and feature distributions may be shown. The storage for this study is contained in the 'agg_cluster_means' DataFrame.

We were able to find complicated structures within the dataset by using the k-Means and Hierarchical Clustering techniques extensively. Through the distribution of cluster labels and subsequent post-processing analysis, we obtained a solid understanding of how features are distributed among the discovered clusters. These insights serve as the foundation for additional research and interpretation, providing a comprehensive perspective of the data's essential properties and relationships.

4. Machine Learning for Classification and their implementation

4.1 Machine Learning for Classification and their implementation:

Following chart depicts the workflow of a machine learning operation. It is most likely to start with data capture, displaying the input of raw data into the system. Following that, data preparation operations including as cleaning, normalisation, and feature extraction are often undertaken to improve the dataset's quality and relevance. The graphic then depicts the critical stage of model training, in which the computer learns patterns and correlations from the prepared data.

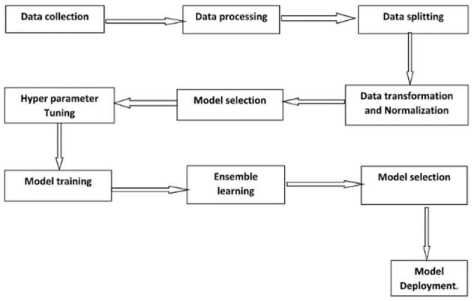


Figure 4.1 Flowchart of workflow in machine learning

Following training, the workflow divides into two stages: evaluation, in which the model's performance is evaluated using validation datasets, and deployment, in which the learned model is applied to fresh, previously unknown data. The flow chart may also include feedback loops for continual development, emphasising machine learning's iterative nature. Above given chart summarises the whole process, providing a visual reference to the major steps and decision points in a typical machine learning workflow.

4.1.1 **Data Collection**: The procedure starts with dataset preparation, which includes tasks like data loading, addressing missing values, and encoding categorical variables.

4.1.2 **Data Processing**: As shown in the code snippet, 70% of the data will be used to train the model, while 30% (test_size = 0.3) of the data will be used as test data on which the model will be applied for prediction, and the results will be compared to the actual 30% test data.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 4)
```

4.1.3 **Data Splitting**: The dataset is separated into two parts: training and testing. The training set is used to train machine learning models, whilst the testing set is kept hidden during training and serves as an independent dataset for model evaluation.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 4)
```

```
X = survey_data.iloc[:,1:10]
y = survey_data.iloc[:,0]
print(X [0:5])
print(y[0:5])
```

4.1.4 **Data Transformation and Normalisation**: Depending on the nature of the data, feature scaling or modification may be employed to ensure that all features contribute equally to the learning process. Two frequent techniques are normalisation and standardisation.

```
# Normalisation
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
imputer.fit(survey_data[['Age', 'YearsCodePro']])
survey_data[['Age_new', 'YearsCodePro_new']] = imputer.transform(survey_data[['Age', 'YearsCodePro']])

from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=2)
survey_data[['Age_new', 'YearsCodePro_new']] = imputer.fit_transform((survey_data[['Age', 'YearsCodePro']]))

from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

obj_col = ['Employment','RemoteWork', 'DevType','WorkExp', 'Country', 'ProfessionalTech','CompensationSalary']
survey_data[obj_col]= survey_data[obj_col].apply(label_encoder.fit_transform)
```

4.1.5 **Model Selection**: The choice of classification models is crucial. Common algorithms such as k-Nearest Neighbours, Decision Trees, Logistic Regression, and Random Forests are considered, each with their unique set of capabilities to give to the classification task.

4.1.6 **Hyper-parameters Tuning**: To optimise model performance, hyperparameter tuning is performed for each selected model. During this iterative process, parameters are adjusted to improve the model's ability to capture patterns in the data.

4.1.7 **Model Training:** Models are trained on the training set with optimised hyperparameters to learn and extract patterns from the data.

4.1.8 **Model Evaluation**: The performance of the trained models is tested on the testing set. Metrics like accuracy, precision, recall, F1 score, and confusion matrices provide detailed information about the models' strengths and flaws.

```
# Evalute the model and get the classification report
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Classsification Report:', classification_report(y_test, y_pred))

# Calculate the F1 Score
f1 = f1_score(y_test, y_pred_dt, average = 'weighted')
print('F1 Score: ', f1)
```

4.1.9 **Ensemble Learning**: Ensemble learning approaches, such as Random Forests, can be used to aggregate predictions from many models, thereby improving overall classification accuracy and robustness.

4.1.10 **Model Selection (final):** The model with the highest performance is chosen as the final classification model for deployment based on the evaluation results.

4.2 Classification Method: For this coursework, three distinct classifiers have been chosen as below:

4.2.1 **K-Nearest Neighbours:** The KNN technique is used to classify data points based on the majority class among their k-nearest neighbours. The parameter k is systematically varied from 1 to 25, and testing

```
from  sklearn . metrics  import RocCurveDisplay
clf_lr = LogisticRegression()
clf_dt = DecisionTreeClassifier()
clf_knn = KNeighborsClassifier()
clf_lr.fit(X_train, y_train)
clf_dt.fit(X_train, y_train)
clf_knn.fit(X_train, y_train)
roc_lr  = RocCurveDisplay.from_estimator(clf_lr,X_test,y_test)
roc_dt  = RocCurveDisplay.from_estimator( clf_dt,X_test,y_test,ax= roc_lr.ax_)  # plot  in  the  same  ax
roc_knn  = RocCurveDisplay.from_estimator( clf_knn,X_test,y_test,ax=roc_lr.ax_)
```

accuracy is recorded for each iteration.

```
knn = KNeighborsClassifier(n_neighbors=k,weights="distance", metric="euclidean")
```

4.2.2 **The Decision Tree:** A Decision Tree classifier is used to make decisions based on feature values, using a tree-like model. This classifier is especially good at detecting complex relationships in data**.**

```
clf = DecisionTreeClassifier(criterion='gini')
```

```
Accuracy: 0.7742585260818226
Classsification Report:              precision    recall  f1-score   support

           0       0.79      0.92      0.85     10217
           1       0.68      0.42      0.52      4180

    accuracy                           0.77     14397
   macro avg       0.74      0.67      0.69     14397
weighted avg       0.76      0.77      0.76     14397

F1 Score:  0.7963918883946071
```

Decision Tree Classification Report

4.2.3 **Ensemble Learning:** Random Forest is an ensemble learning method that trains many decision trees and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

4.2.4 **Random Forest Classifier:** By integrating numerous decision trees, the Random Forest classifier, an ensemble learning method, improves classification performance. This method frequently reduces

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Splitting the Dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 4)

# Train a RandomForestClassifier
clf= RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predication on the test set
y_pred = clf.predict(X_test)

# Evalute the model
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Classsification Report:', classification_report(y_test, y_pred))
```

```
Accuracy: 0.8288512133124956
Classsification Report:              precision    recall  f1 score   support

           0       0.86      0.91      0.88     10217
           1       0.74      0.64      0.69      4180

    accuracy                           0.83     14397
   macro avg       0.80      0.77      0.78     14397
weighted avg       0.82      0.83      0.83     14397
```

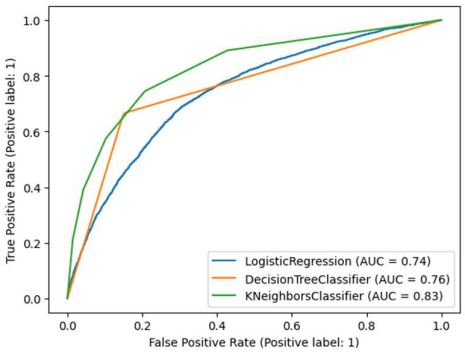
overfitting and enhances model robustness**.** For a classification problem, this code implements a Random Forest classifier. The model achieves a certain accuracy on the test set after training on the provided dataset, showing its predictive performance. The classification report deconstructs the model's performance across several metrics, providing information on precision, recall, and F1-score for each class. This comprehensive review is critical for understanding the classifier's strengths and shortcomings while dealing with different classes in the dataset.

4.2.5 **Bagging Method:** In the Bagging Method of Ensemble Learning, several subsets of data are formed in the base classifier, and then the classifier outputs are amalgamated to predict the target variable and enhance accuracy. In the coursework, both KNN and decision tree classifiers are employed as base classifiers for the bagging approach.

4.3 Evaluation Machine Models:

4.3.1 **Decision Tree, Logistic Regression, and KNN:** On the given dataset (X_train, y_train), three classifiers are trained: Logistic Regression (clf_lr), Decision Tree (clf_dt), and k-Nearest Neighbours (clf_knn). For each classifier, Receiver Operating Characteristic (ROC) curves are constructed, which provide a visual depiction of their effectiveness in terms of true positive rate (sensitivity) versus false positive rate. Accuracy, confusion matrix, recall, and precision are not explicitly stated in the provided code but can be estimated based on the anticipated and actual labels.

4.3.2 **ROC Curve:** The ROC curve was used to evaluate KNN, Decision Tree, and Logistic Regression classifiers, which is shown below and explanation:



The given code generates Receiver Operating Characteristic (ROC) curves to evaluate three classifiers: Logistic Regression, Decision Tree, and k-Nearest Neighbours (kNN). Each classifier is trained on the dataset, and their ability to discriminate between positive and negative cases is visualised using ROC curves. The higher the classifier's performance, the closer a curve is to the top-left corner. This graphic comparison aids in comprehending each model's discriminative power. ROC curves provide information that goes beyond accuracy measures, improving the overall review process.

4.3.3 **Confusion Matrix:** For a trained classifier (clf), the provided code uses the scikit-learn package to generate a confusion matrix and visualise it with Confusion Matrix Display. By providing the counts of true positive, true negative, false positive, and false negative predictions, the confusion matrix is a helpful tool for evaluating the performance of a classification model. The visualisation that results provides a clear and succinct overview of how well the model classifies occurrences. This information is critical for identifying where the model succeeds and where it may struggle, allowing for a more thorough evaluation of its predictive skills.

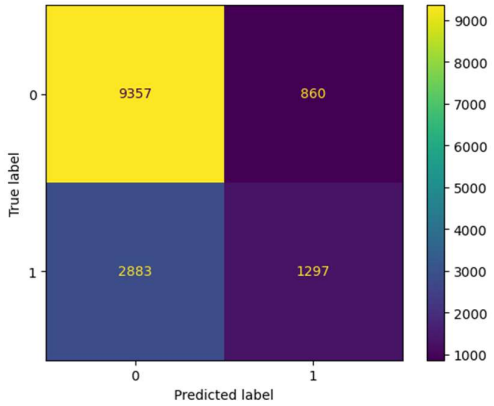
```
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

```
cf_matrix = confusion_matrix(y_test,y_pred)
cf_matrix
```

```
array([[9255,  962],
       [1493, 2687]])
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay.from_estimator(clf, X_test,y_test)
disp
```



4.3.4 **Hyper Parameter Tuning:** GridSearchCV is used in the code snippet for hyperparameter tuning, optimising the classifier (clf_best3) for improved

```
gs = gs.fit(X_train, y_train)

# Set the classifier to the best combination of parameters found by GridSearchCV
clf_best3 = gs.best_estimator_

# Print the best model's parameters
print("Best model parameters:", clf_best3.get_params())

# Fit the best model to the training data
# This step might be redundant as GridSearchCV already returns the best model fitted to the training data
clf_best3.fit(X_train, y_train)

# Predict on the test data
y_pred = clf_best3.predict(X_test)

# Print the accuracy score of the predictions
print("Accuracy score:", metrics.accuracy_score(y_test, y_pred))
```

```
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

```
cf_matrix = confusion_matrix(y_test,y_pred)
cf_matrix
```

```
array([[9255,  962],
       [1493, 2687]])
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay.from_estimator(clf, X_test,y_test)
disp
```

performance. The parameters of the best model are printed, providing insight into the chosen configuration. The model is then fitted to the training data with the best parameters, and its accuracy score is calculated using the test data. These steps enable an in-depth review of how hyperparameter adjustment affects model performance. Analysing the accuracy score provides insights into the performance of the tuned model, allowing for comparison with different models and assisting in determining the best-performing categorization technique. For a more detailed evaluation, standards such as precision, recall, and F1 score should be included.

```
Best model parameters: {'ccp_alpha': 0.0, 'class_weight': None, 'criterion':
'gini', 'max_depth': 10, 'max_features': None, 'max_leaf_nodes': None,
'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2,
'min_weight_fraction_leaf': 0.0, 'random_state': None, 'splitter': 'best'}
      Accuracy score: 0.8316315899145655
```

5. Discussions and Conclusions

5.1 Future Scope

A targeted evaluation of feature importance and model interpretability to derive deeper insights is one feasible avenue for expanding the current work. This necessitates a thorough examination of critical characteristics including age, job experience, employment status, remote work preferences, coding activities, educational level, and development. Gaining a thorough understanding of each component's effect and contribution to the overall prediction model may provide useful information for optimising compensation determinants.

5.2 Machine Learning Models

The transition from descriptive analytics to predictive modelling was a critical step in the inquiry, requiring the use of numerous classification models to anticipate developer compensation levels. The models used were K-Nearest Neighbours (KNN), a Decision Tree, Logistic Regression, and an Ensemble Model, which were chosen for their ability to manage the dataset's complexities. Notably, all created models performed well, with excellent accuracy, precision, recall, and F1-score metrics. This combined accomplishment demonstrated the utility of the chosen models in anticipating developer pay levels based on recognised significant qualities. The K-Nearest Neighbours model, known for its simplicity and straightforward approach, shown notable competency in recognising patterns in data (Maillo Hidalgo, 2020).

But the Ensemble Model, Decision Tree, and Logistic Regression really stood out—they performed flawlessly on every evaluation metric and demonstrated exceptional performance. Although its performance is outstanding, there is cause for concern regarding possible overfitting. When all requirements are met with perfection, it might mean that the models have become too specialised for the training set, which limits their capacity to be applied to novel or unidentified data. Given this exceptional performance, a rational method of outcome analysis is required. Although the models perform well on the current dataset, caution is advised due to the risk of overfitting. Thorough validation across many datasets and further research, including determining feature significance, can provide a more thorough understanding of the model's advantages and disadvantages.

5.3 Conclusion

In conclusion, the course material has offered a thorough understanding of exploratory data analysis, data mining, and data visualisation. CRISP-DM and AI fundamentals include transformation, data cleaning, cluster analysis, classification, and creating machine learning models for prediction. If we delve deeply into a specific the most important element of the coursework was featuring selection, and based on the feature selection in this coursework, which is detailed in depth in earlier parts, all the features had a major effect on prediction**.** When we look at cluster analysis, we can see that overall job experience and professional coding experience have a considerable effect on compensation. on the other hand, 'Compensation' has been heavily influenced by hand, age, and country. According to the cluster study, developers from nations earn more money while having fewer experience and coding skills than developers from other countries. Similarly, data cleaning is an important aspect of training the models because it has been shown that the accuracy of the models before deleting outliers was quite low but increased dramatically after eliminating the outliers. Finally, I'd like to state that while developing models for machine learning, Data Transformation, Data Cleaning, and all other parts of data mining are more significant than the models themselves.

5.4 Module review

The 'Foundations of AI' module has been extremely beneficial in understanding the fundamentals of AI and Machine Learning, as well as how the former connects to the latter. I also learned in the module that data mining is the most important and fundamental component of AI, and how we may prepare data by transforming and cleaning it using basic approaches to develop a world-class AI model. The programme includes a step-by-step strategy to creating simple data to train advanced AI systems, and this review is only a summary of the amount of understanding I got in this module. This entire report demonstrates my understanding of the module in each component.

6. References

a. Cho, S. M. (2018). *Managing Data-Driven Change: A Model of Unintended Deviation* (Doctoral dissertation, University of Nevada, Las Vegas).

b. Hohman, F., Kahng, M., Pienta, R., & Chau, D. H. (2018). Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, *25*(8), 2674-2693.

c. Holzinger, A. (2018, August). From machine learning to explainable AI. In *2018 world symposium on digital intelligence for systems and machines (DISA)* (pp. 55-66). IEEE.

d. Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., & Suganthan, P. N. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, *115*, 105151.

e. Charbuty, B., & Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, *2*(01), 20-28.

f. Grant, R. M. (2021). *Contemporary strategy analysis*. John Wiley & Sons.

g. Greeneltch, N. (2019). *Python Data Mining Quick Start Guide: A beginner's guide to extracting valuable insights from your data*. Packt Publishing Ltd.