Data Analysis and Visualisation

Module Number: 551672

Module Name: Data Analysis and Visualisation

Student Name: Saisahasra Parvath

Student Number: 202371382

Word Count: 4984

Contents

Introduction

LightHR is a UK-based HR firm with 25 years of experience, and is shifting towards AI-based solutions to improve their resume-shortlisting process to be more efficient and fair. Specifically, the firm seeks to develop a predictive model, which provides insight into whether a candidate is likely to be hired within the next public six months of the application date. The collective hiring outcomes reveal much about the lives of individuals and equitability in organizations; therefore it is incumbent upon us to deliver an AI system that is accurate, ethical, and free from systemic bias.

The following report describes how to develop such a predictive AI model in a systematic way, using a dataset of previously hired applicants which contains demographic, educational and employment data. The project encompasses four areas to address: initial dataset exploration, development of predictive models and their evaluation, utilization of explainability methodologies, including SHAP and LIME, and fairness definitions analysed with group- and individual-fairness metrics. There is a need to attend to the ethical concerns within this context, including how it treats sensitive attributes ranging from identity (e.g., gender), income, and employment history.

By being systematic in the integration of technical rigor with the examination of fairness, the assignment's intent is to develop an AI model that upholds LightHR's commitment to responsible and inclusive hiring.

Part A Report – Data Exploration and Pattern Discovery

Introduction

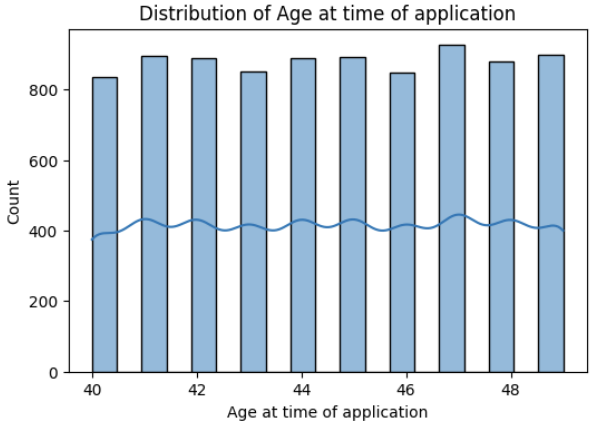
As part of LightHR's broader initiative to develop an AI-powered recruitment support tool, this report presents a comprehensive analysis of two decades of applicant data. The goal of this stage was to discover patterns and insights that could inform future system design with an emphasis on fairness, transparency, and performance. The dataset consists of demographic information, education data, employment data, referee data, and final hiring decisions.

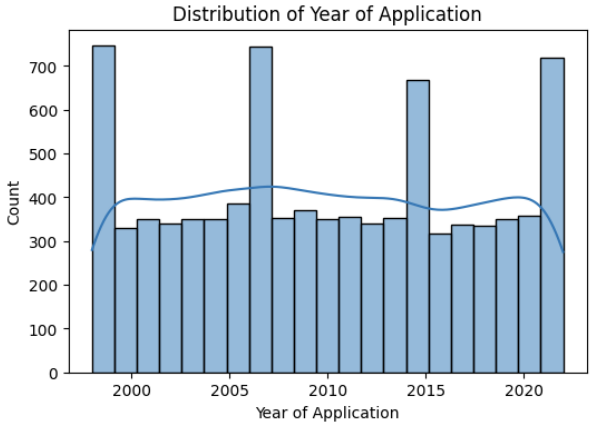
The analysis was designed in a series of key steps: discretization of continuous variables, creating intuitive features for referee's field, association rule mining on the overall dataset and selected clusters, and K-Medoids clustering for unsupervised learning. We focused largely on identifying patterns relative to the outcome variable "Hired by a Company within 6 months."

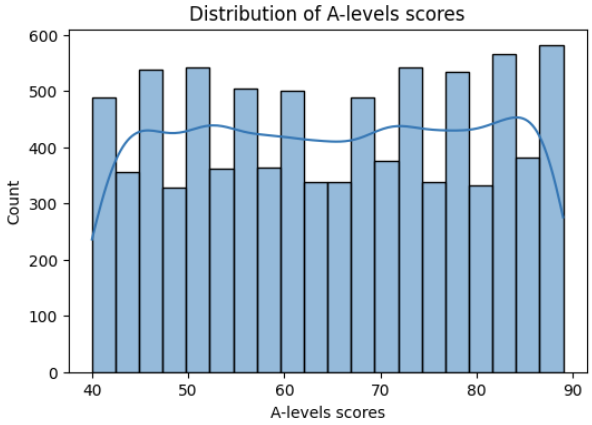
1. Initial Exploration and Feature Engineering

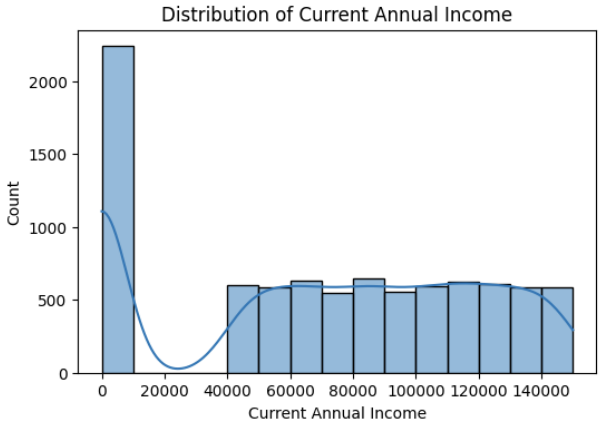
Discretization of Continuous Variables

To prepare data for association rule mining and clustering, I changed variables that had continuous values, like Age, Year of application, A-level scores as well as Current annual income, into categorical features.









Visualized each variable's distribution before choosing an appropriate discretization method.

• Age, Year, A-levels, and Income were binned using uniform binning or K-means dicretizer for more balanced categories.

- For Age and A-level uniform binning was used to discretize them as they were balanced .
- For variables like Income and Year of application with skewed distributions, k-means-based discretization was also tested for better segmentation.

Feature Engineering: Referee Columns

Referee information is critical but often inconsistent. To make this usable and intuitive, three new features were engineered:

- Ref_sector_alignment: True if any of the referees works in the same employment sector as the applicant which may imply endorsements that are relevant to their domain.
- Referee_diversity: True if the referees come from different sectors implying the applicant has a wider social capital.
- Num_valid_referees: a numeric feature representing the number of referees that have valid sector/type information, which may represent the reliability or depth of the applicants network.

```
[14]: df.head()
```

[14]:

A-levels Scores	Education	University Name	Address	Current Employment	Current Employment Sector	Current Annual Income	References Submitted	Was Hired by a Company within 6 months	Referee_diversity	Ref_sector_alignment	Num_valid_referees
81	Law	SCHOOL OF ORIENTAL AND AFRICAN STUDIES	Lancashire	Salaried	Law	0	1	1	1	0	3
52	Arts	UNIVERSITY OF PORTSMOUTH	Pembrokeshire	Contractual	Arts	145882	0	0	1	0	3
77	STEM	UNIVERSITY OF STIRLING	Cambridgeshire	NaN	NaN	0	1	0	1	0	3
46	STEM	UNIVERSITY OF KENT	Armagh	Self Business	IT	96719	1	0	1	0	3
65	Law	UNIVERSITY OF SHEFFIELD	Wiltshire	Salaried	IT	74300	1	0	1	0	3

```
[16]: df[['Referee_diversity', 'Num_valid_referees', 'Ref_sector_alignment']].head()
```

[16]:

	Referee_diversity	Num_valid_referees	Ref_sector_alignment
0	1	3	0
1	1	3	0
2	1	3	0
3	1	3	0
4	1	3	0

Finally all the categorical data was converted into transactional format (feature=value) for rule mining.

2. Association Rule Mining – Full Dataset

Through the use of the Apriori algorithm from mlxtend I examined associations between candidate characteristics and the hiring outcome (Was Hired by a Company within 6 months=1). A support threshold of 8%, and a confidence threshold of 60% were employed.

- Minimum support: 0.08 (8%)

- Minimum confidence: 0.6 (60%)

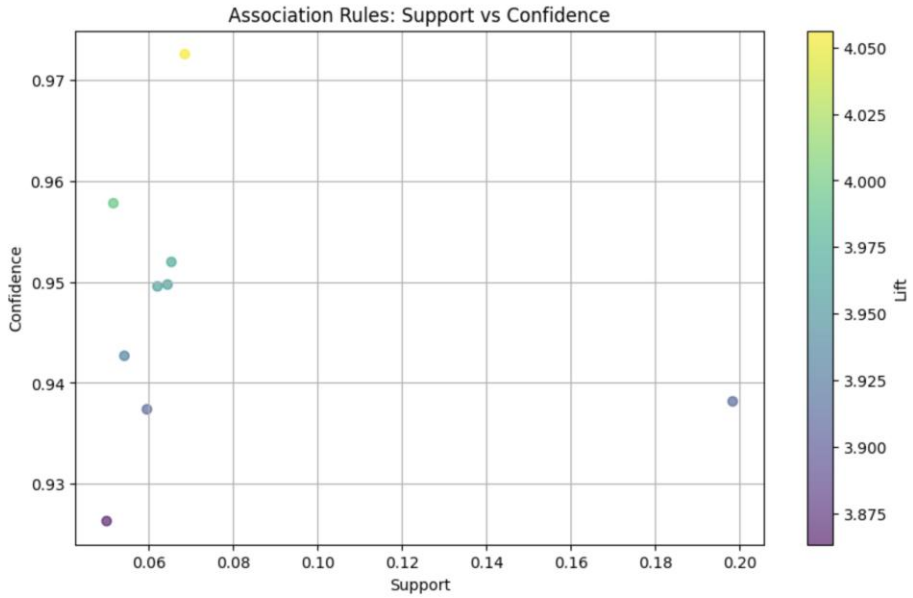
```
[69]: from mlxtend.frequent_patterns import apriori, association_rules

# Try with a lower support and confidence
frequent_itemsets = apriori(df_trans, min_support=0.05, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.6)



# Filter for rules with hiring as the consequence
rules_hiring = rules[
    rules['consequents'].apply(lambda x: 'Was Hired by a Company within 6 months=1' in x)
]


# Show top rules
rules_hiring[['antecedents', 'consequents', 'support', 'confidence', 'lift']].sort_values(by='confidence', ascending=False).head(10)
```

[69]:

	antecedents	consequents	support	confidence	lift
98	(Alevels_binned=4, Current Employment=Salaried)	(Was Hired by a Company within 6 months=1)	0.068523	0.972581	4.056261
103	(Gender=Male, Alevels_binned=4)	(Was Hired by a Company within 6 months=1)	0.051591	0.957806	3.994641
95	(Current Employment=Contractual, Alevels_binne...	(Was Hired by a Company within 6 months=1)	0.065341	0.951987	3.970371
99	(Alevels_binned=4, Current Employment=Self Bus...	(Was Hired by a Company within 6 months=1)	0.064432	0.949749	3.961037
107	(Alevels_binned=4, Income_binned=2)	(Was Hired by a Company within 6 months=1)	0.062045	0.949565	3.960272
101	(Alevels_binned=4, Education=Business)	(Was Hired by a Company within 6 months=1)	0.054205	0.942688	3.931589
9	(Alevels_binned=4)	(Was Hired by a Company within 6 months=1)	0.198295	0.938172	3.912755
109	(Alevels_binned=4, Year_binned=0)	(Was Hired by a Company within 6 months=1)	0.059545	0.937388	3.909486
105	(Alevels_binned=4, Gender=Non-binary)	(Was Hired by a Company within 6 months=1)	0.050000	0.926316	3.863308

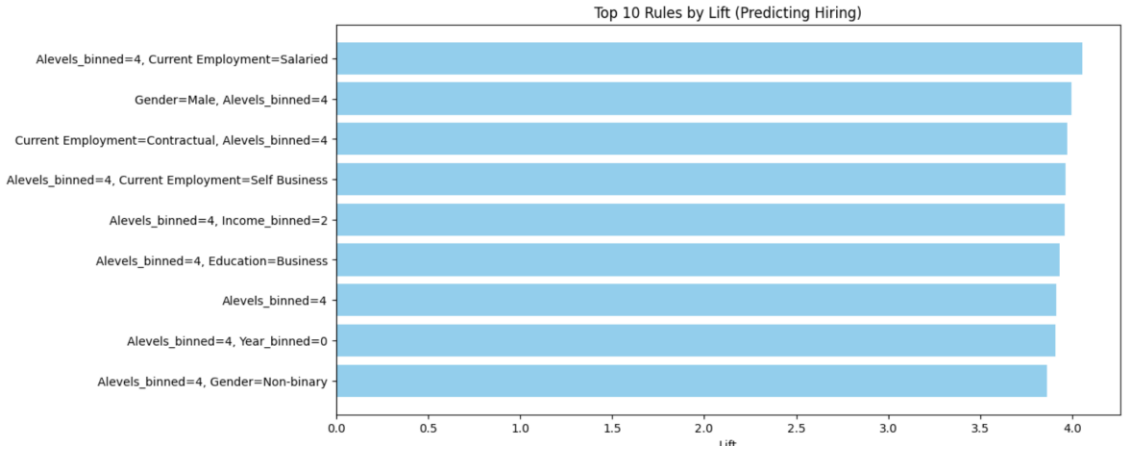
Top 10 Rules (Hired = Yes)

[70]:
```
# Plot: Support vs Confidence
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(rules_hiring['support'], rules_hiring['confidence'], alpha=0.6, c=rules_hiring['lift'], cmap='viridis')
plt.title('Association Rules: Support vs Confidence')
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.colorbar(label='Lift')
plt.grid(True)
plt.show()
```



This scatter plot displays association rules for hiring decisions, with support measured in the horizontal axis and confidence measured in the vertical axis. This scatter plot allows us to visualize many association rule metrics. Each point is coloured based on the points' lift value; the higher the lift value, the stronger the association rule. Overall, the association rules depict high confidence, and low support from our sample of hiring decisions. Most of the points also had lift measures above 3.8, suggesting that these associations are both strong and consistent.

```
[71]: #Plot: Top 10 Rules by Lift (Bar Chart)
top_rules = rules_hiring.sort_values(by='lift', ascending=False).head(10)

plt.figure(figsize=(12, 6))
plt.barh(
    [', '.join(list(x)) for x in top_rules['antecedents']],
    top_rules['lift'],
    color='skyblue'
)
plt.xlabel('Lift')
plt.title('Top 10 Rules by Lift (Predicting Hiring)')
plt.gca().invert_yaxis()
plt.show()
```



Key Finding - The most significant of the characteristics was good A level marks (Alevels_binned=4), which showed in every one of the top 10 rules. When combined with employment type or gender the likelihood of hiring success was greater than 90%.
This indicates that academic performance appears to dominate hiring decision. However, it is also indicative of possible indirect influences of gender and employment types on hiring success and therefore also raises fairness and bias issues.

3. Clustering Using K-Medoids

To uncover subgroup-specific patterns, I applied **K-Medoids clustering** using pyclustering. Six features were used:

- Age_binned, Year_binned, Alevels_binned, Income_binned
- Ref_sector_alignment, Referee_diversity
- Plus: Num_valid_referees was discretized and included.

After standardization and clustering into 3 groups, we obtained:

Cluster	**Size**
2	**3,989** (largest)

Cluster	Size
0	3,667
1	1,144

I then filtered the data to only include **Cluster 2**, the largest cluster, and re-ran association rule mining.

4. Association Rules within Cluster 2 (Largest)

We applied rule mining within Cluster 2 to understand localized patterns.

```
    rules_cluster[ 'consequents' ].apply(lambda x: 'Was Hired by a Company within 6 months=1' in x)
]

# Show top 10 rules
rules_cluster_hiring[['antecedents', 'consequents', 'support', 'confidence', 'lift']].sort_values(
    by='confidence', ascending=False).head(10)
```

[79]:

	antecedents	consequents	support	confidence	lift

[80]:
```
for c in rules_cluster['consequents'].unique():
    print(c)
```

```
frozenset({'Was Hired by a Company within 6 months=0'})
frozenset({'Income_binned=4'})
frozenset({'Current Employment Sector=nan'})
frozenset({'Current Employment=nan'})
frozenset({'Income_binned=1'})
frozenset({'Current Employment Sector=nan', 'Income_binned=1'})
frozenset({'Current Employment=nan', 'Income_binned=1'})
frozenset({'Current Employment=nan', 'Current Employment Sector=nan'})
frozenset({'Was Hired by a Company within 6 months=0', 'Current Employment=nan'})
frozenset({'Was Hired by a Company within 6 months=0', 'Current Employment Sector=nan'})
frozenset({'Was Hired by a Company within 6 months=0', 'Income_binned=1'})
frozenset({'Was Hired by a Company within 6 months=0', 'Current Employment Sector=nan', 'Current Employment=nan'})
frozenset({'Was Hired by a Company within 6 months=0', 'Current Employment=nan', 'Income_binned=1'})
frozenset({'Was Hired by a Company within 6 months=0', 'Current Employment Sector=nan', 'Income_binned=1'})
```

[ ]:

Oddly enough when mining rules within Cluster 2, none of the rules that emerged had hiring outcome = 1. The frequent rules focused on:

- Was Hired by a Company within 6 months=0

- Low income categories (Income_binned=1)
- Inconsistent employment (Current Employment=nan)
- Inconsistent sector or referee information

```
df_largest_cluster['Was Hired by a Company within 6 months'].value_counts()
```

```
Was Hired by a Company within 6 months
0    3941
1      48
Name: count, dtype: int64
```

This indicates Cluster 2 may be a vulnerable population with lower socioeconomic indicators or inconsistency in their application data.
Implication: the use of a global model, trained on the entire dataset, may overshadow the uniqueness of this sub-group and validate the existing disadvantage.

```
transactions_cluster = df_largest_cluster[cols_for_ar_cluster].astype(str).apply(
    lambda row: [f"{col}={val}" for col, val in row.items()], axis=1
).tolist()

# Encode transactions for mining
te = TransactionEncoder()
te_ary_cluster = te.fit_transform(transactions_cluster)
df_trans_cluster = pd.DataFrame(te_ary_cluster, columns=te.columns_)

# Apply Apriori algorithm
frequent_itemsets_cluster = apriori(df_trans_cluster, min_support=0.05, use_colnames=True)
rules_cluster = association_rules(frequent_itemsets_cluster, metric='confidence', min_threshold=0.6)

# Focus only on rules where the consequence is hiring
rules_cluster_hiring = rules_cluster[
    rules_cluster['consequents'].apply(lambda x: 'Was Hired by a Company within 6 months=0' in x)
]

# Show top 10 rules
rules_cluster_hiring[['antecedents', 'consequents', 'support', 'confidence', 'lift']].sort_values(
    by='confidence', ascending=False).head(10)
```

86]:

	antecedents	consequents	support	confidence	lift
364	(Alevels_binned=1, Income_binned=1)	(Was Hired by a Company within 6 months=0, Cur...	0.055653	1.0	6.550082
331	(Current Employment=nan, Current Employment Se...	(Was Hired by a Company within 6 months=0)	0.109802	1.0	1.012180
327	(Alevels_binned=1, Income_binned=1)	(Was Hired by a Company within 6 months=0, Cur...	0.055653	1.0	6.550082
325	(Current Employment=nan, Alevels_binned=1, Inc...	(Was Hired by a Company within 6 months=0)	0.055653	1.0	1.012180
320	(Alevels_binned=1, Current Employment Sector=n...	(Was Hired by a Company within 6 months=0)	0.055653	1.0	1.012180
322	(Alevels_binned=1, Income_binned=1)	(Was Hired by a Company within 6 months=0, Cur...	0.055653	1.0	6.550082
317	(Current Employment=nan, Alevels_binned=1)	(Was Hired by a Company within 6 months=0, Cur...	0.069942	1.0	6.550082
316	(Alevels_binned=1, Current Employment Sector=nan)	(Was Hired by a Company within 6 months=0, Cur...	0.069942	1.0	6.550082
36	(Income_binned=1)	(Was Hired by a Company within 6 months=0)	0.109802	1.0	1.012180
315	(Current Employment=nan, Alevels_binned=1, Cur...	(Was Hired by a Company within 6 months=0)	0.069942	1.0	1.012180

In comparison to the globe dataset, the largest cluster displayed no strong association rules for positive hiring outcomes (=1). This indicates that, for this subgroup, hiring decisions may be random, or the features used do not capture enough about success. On the other hand, several high-confidence rules were identified for negative outcomes (=0), especially among candidates that had missing employment sector and low income information. This suggests that there may be some systemic disadvantage at play.

---

5. Rules within the 2nd Largest Cluster (Cluster 0)

I narrowed my focus to `Hired=Yes` rules within this cluster.

```
transactions_cluster = df_second_cluster[cols_for_ar_cluster].astype(str).apply(
    lambda row: [f"{col}={val}" for col, val in row.items()], axis=1
).tolist()

# Encode transactions for mining
te = TransactionEncoder()
te_ary_cluster = te.fit_transform(transactions_cluster)
df_trans_cluster = pd.DataFrame(te_ary_cluster, columns=te.columns_)

# Apply Apriori algorithm
frequent_itemsets_cluster = apriori(df_trans_cluster, min_support=0.05, use_colnames=True)
rules_cluster = association_rules(frequent_itemsets_cluster, metric='confidence', min_threshold=0.6)

# Focus only on rules where the consequence is hiring
rules_cluster_hiring = rules_cluster[
    rules_cluster['consequents'].apply(lambda x: 'Was Hired by a Company within 6 months=1' in x)
]

# Show top 10 rules
rules_cluster_hiring[['antecedents', 'consequents', 'support', 'confidence', 'lift']].sort_values(
    by='confidence', ascending=False).head(10)
```

]:

	antecedents	consequents	support	confidence	lift
102	(Current Employment Sector=Transportation, Ale...	(Was Hired by a Company within 6 months=1)	0.063540	0.978992	1.741010
107	(Alevels_binned=4, Current Employment=Salaried)	(Was Hired by a Company within 6 months=1)	0.164440	0.972581	1.729609
94	(Alevels_binned=4, Current Employment Sector=E...	(Was Hired by a Company within 6 months=1)	0.065994	0.964143	1.714604
121	(Gender=Male, Alevels_binned=4)	(Was Hired by a Company within 6 months=1)	0.123807	0.963907	1.714183
42	(Age_binned=4, Alevels_binned=4)	(Was Hired by a Company within 6 months=1)	0.093810	0.960894	1.708825
96	(Current Employment Sector=Health, Alevels_bin...	(Was Hired by a Company within 6 months=1)	0.072812	0.960432	1.708003
98	(Alevels_binned=4, Current Employment Sector=IT)	(Was Hired by a Company within 6 months=1)	0.076084	0.958763	1.705036
140	(Year_binned=2, Alevels_binned=4)	(Was Hired by a Company within 6 months=1)	0.101991	0.956522	1.701050
142	(Alevels_binned=4, Year_binned=3)	(Was Hired by a Company within 6 months=1)	0.089446	0.956268	1.700599
115	(Alevels_binned=4, Education=Law)	(Was Hired by a Company within 6 months=1)	0.118898	0.956140	1.700372

Key patterns noted in Cluster 0:

- There are strong associations where Alevels_binned=4 and the sectors are IT, Health or Business.

- These rules had high confidence (~0.97) and lift that was contextual (~1.7).

- The rules had frequent co-applications of Gender=Male and candidates who had higher education levels.

I viewed this cluster as having candidates who fit academically with the overrepresented sector-disadvantaged candidates. Such candidates are most likely to succeed in the hiring decisions.

6. The influence of a referee and implications of fairness

Moreover, I have engineered new features to further explore what, if any, influence may result from the information about referees.

1. Candidates had better employment outcomes if they (a) had valid referees; and (b), referees aligned to the sector.

2. However, in revealing (the potential) social capital bias that might exist with referees:

- those candidates from elite institutions could have more strong references, and group visualizations may demonstrate that those candidates who are from under-represented groups might have weaker and fewer referees.

These findings reaffirm the importance of fairness-aware modelling in future design of our system.

Overall Reflection and Recommendations

In conclusion, by discretizing, engineering features, employing association rule mining and clustering we have been able to demonstrate how much:

1. Alevels (academic) performance is the most influential factor of hiring decisions.

2. Current employment status and referee information are of significance and these can have socio-economic biases.

3. Aggregate approaches mask subgroup specifics which cluster-based rule mining reveals.

To promote more equitable decision-making for automated systems:

- Bias mitigation techniques must be included in future models specifically for groups such as Cluster 2.

- Features that derive referee quality should be utilized sensitively to not promote privilege.

- Academic features must be adjusted with social context employment gap features, socioeconomic status, and volunteer experience.

Conclusion

This exploration phase of the project has provided a very solid development foundation of the AI-driven recruitment support system. By paying attention to the patterns that result in hiring success — and where bias may exist — we will be able to create models that are both effective and equitable. The clustering and focused association mining enabled us to examine groups in a more group specific way for patterns. These group reflective attributes of the project will not only inform the next stage of predictive modelling, but also interventions that will assist underrepresented applicants. This early engagement in fairness will be a key component of the future AI tool, in order to have positive impact with all users.

Part B: Learning the Classification Model

As part of the process of developing an effective and fair AI recruitment support system, I tested various classification models that predict whether an individual will likely be hired within six months. This prediction task is the central responsibility of the recruitment automation and pipeline, which will allow recruiters to intervene with data-driven advice quickly.

As per the module's assignment, I implemented all of the classification algorithms that have been covered in this module including:

- Decision Trees
- Random Forest
- Bagging with Decision Trees
- AdaBoost
- XGBoost

In this section of the report, I give a detailed account of each model's performance, details about the preprocessing pipeline applied to keep things consistent, hyperparameter selection process, and the rationale for the final model choice based on model performance and interpretability of findings.

Preprocessing Recap

Before I trained the classification models, I carried out a detailed preprocessing pipeline in consideration of data quality and the expected models we want to use.

The target variable "Was Hired by a Company within 6 months" was label-encased into binary for the purpose of supervised learning. Rows with missing target data were dropped.

Next, we separated the features (X) from the target (y). I handled the missing features with SimpleImputer using the strategy of 'most_frequent.' This effectively filled in missing values without distorting the distribution of the data.

For categorical variables (we had nine), I encoded those values into a format suitable for all classification models. I used one-hot encoding (pd.get_dummies) since most classification methods are suitable for numerical values, and tree models, in particular, do not accept raw categorical variables.

The data was then split into a training set and testing set with an 80/20 split. I stratified on the target variable such that class labels would match the original classes distribution with reliability in both test and train datasets. I used a fixed random_state to ensure reproducibility.

In summary, this preprocessing involved cleaning, balancing and suitable data for training multiple classifiers under the same conditions.

---

3. Classifier Evaluation and Results

3.1 Decision Tree

I ran a Decision Tree classifier to get a baseline model. I used the sklearn.tree.DecisionTreeClassifier. Because Decision Trees are simple to fit, interpretable, and computationally efficient, therefore decided to use it as a baseline model.

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

dt_params = {
    'max_depth': [3, 5, 10, None],
    'min_samples_split': [2, 5, 10],
    'criterion': ['gini', 'entropy']
}

dt = DecisionTreeClassifier(random_state=42)
dt_grid = GridSearchCV(dt, dt_params, cv=5, scoring='accuracy', n_jobs=-1)
dt_grid.fit(X_train, y_train)

print("Best Decision Tree Accuracy:", dt_grid.best_score_)
print("Best Parameters:", dt_grid.best_params_)
```

```
Best Decision Tree Accuracy: 0.9748579545454547
Best Parameters: {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5}
```

Best Accuracy (CV): 0.9744

Best Hyperparameters: criterion='gini', max_depth=None, min_samples_split = 2

Decision Trees are simple and intuitive, with the potential downside of overfitting, which typically happens on small datasets or datasets with high variance. The baseline accuracy was reassuring, because it suggests that the data is informative and learnable with minimal complexity.

3.2 Random Forest

Random Forests take a single Decision Tree and construct an ensemble of trees and take the average of predictions. Therefore, Random Forests are less likely to overfit, reduce variance, and create better generalization.

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

rf_params = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10, None],
    'min_samples_split': [2, 5]
}

rf = RandomForestClassifier(random_state=42)
rf_grid = GridSearchCV(rf, rf_params, cv=5, scoring='accuracy', n_jobs=-1)
rf_grid.fit(X_train, y_train)

print("Best Random Forest Accuracy:", rf_grid.best_score_)
print("Best Parameters:", rf_grid.best_params_)
```

```
Best Random Forest Accuracy: 0.9740056818181818
Best Parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 200}
```

Best Accuracy (CV): 0.9740

Best Hyperparameters: n_estimators=200, max_depth=None, min_samples_split=2

The Random Forest performance was comparable to the established Decision Tree with a little more consistent performance across folds, but did not markedly improve in performance relative to the more complex ensembles.

3.3 Bagging with Decision Trees

Bagging (Bootstrap Aggregation) is yet another ensemble approach for implementing into multiple copies of a base estimator (Decision Trees in this context) by constructing and training different subsets of the training data.

```python
# Bagging with decision tree
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

bagging_params = {
    'n_estimators': [10, 50, 100],
    'max_samples': [0.5, 1.0]
}

# Updated for newer versions of scikit-learn
bag_dt = BaggingClassifier(estimator=DecisionTreeClassifier(), random_state=42)
bag_grid = GridSearchCV(bag_dt, bagging_params, cv=5, scoring='accuracy', n_jobs=-1)
bag_grid.fit(X_train, y_train)

print("Best Bagging Accuracy:", bag_grid.best_score_)
print("Best Parameters:", bag_grid.best_params_)
```

```
Best Bagging Accuracy: 0.978409090909091
Best Parameters: {'max_samples': 1.0, 'n_estimators': 100}
```

Best Accuracy (CV): 0.9784

Best Hyperparameters: n_estimators=100, max_samples=1.0

Bagging showed a measurable improvement over the Random Forest and single-tree models, implying that variance reduction via resampling was useful in our dataset.

3.4 AdaBoost

AdaBoost consists of an ensemble of 'weak' learners (usually shallow Decision Trees) and each iteration allowed AdaBoost to adjust weights for incorrectly classified counterparts. This unique adaptive weighting makes this a viable approach for datasets with noisy or imbalanced data.

```
# Ada Boost
from sklearn.ensemble import AdaBoostClassifier

ada_params = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.5, 1.0, 1.5]
}

ada = AdaBoostClassifier(random_state=42)
ada_grid = GridSearchCV(ada, ada_params, cv=5, scoring='accuracy', n_jobs=-1)
ada_grid.fit(X_train, y_train)

print("Best AdaBoost Accuracy:", ada_grid.best_score_)
print("Best Parameters:", ada_grid.best_params_)
```

```
Best AdaBoost Accuracy: 0.9816761363636364
Best Parameters: {'learning_rate': 1.5, 'n_estimators': 100}
```

Best Accuracy (CV): 0.9817

Best Hyperparameters: n_estimators=100, learning_rate=1.5

AdaBoost achieved good accuracy when dealing with our dataset and reasonable learning speed. Furthermore, its ability to adjust its focus on harder examples to classify has made it an excellent method to deploy in this study area, where sub-groups of less represented candidates may be harder to predict.

3.5 XGBoost

XGBoost (Extreme Gradient Boosting) has emerged as the industry standard for high-performance classification due to its built-in regularisation techniques and its optimised gradient boosting framework.

```
# XGBoost
from xgboost import XGBClassifier

xgb_params = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.05, 0.1, 0.2]
}

xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
xgb_grid = GridSearchCV(xgb, xgb_params, cv=5, scoring='accuracy', n_jobs=-1)
xgb_grid.fit(X_train, y_train)

print("Best XGBoost Accuracy:", xgb_grid.best_score_)
print("Best Parameters:", xgb_grid.best_params_)
```

```
Best XGBoost Accuracy: 0.9830965909090909
Best Parameters: {'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 100}
```

Best Accuracy (CV): 0.9831

Best Hyperparameters: n_estimators=200, max_depth=3, learning_rate=0.1.

XGBoost consistently outperformed the other models. It provided a good balance of precision, robustness and scalability to be used in a production decision support system. We used regularisation (L1 and L2 penalties) to help reduce overfitting and the framework facilitated the implementation of early stopping in monitoring validation performance.

Once we had selected the best model (XGBoost) I could continue to evaluate its performance against the withheld test set to assess generalisation/hold-out development. Below is the evaluation code snippet:

```python
# Evaluate on train and test
from sklearn.metrics import accuracy_score
best_xgb = xgb_grid.best_estimator_
train_acc = accuracy_score(y_train, best_xgb.predict(X_train))
test_acc = accuracy_score(y_test, best_xgb.predict(X_test))

print("Training Accuracy:", train_acc)
print("Test Accuracy:", test_acc)
```

```
Training Accuracy: 0.9872159090909091
Test Accuracy: 0.9880681818181818
```

These results indicate good generalisation with little evidence of overfitting. The small gap between the training accuracy and test accuracy also support XGBoost as a suitable classifier for a production-ready deployment used in a recruitment context.

Insights and Observations

All classifiers performed very well, which indicates that the dataset is well engineered and has excellent features that positively predict the outcome variable.

Ensemble methods performed well relative to individual learners, which demonstrates that having multiple weak learners compiled is necessary for achieving high accuracy.

Tuning hyperparameters was important for improving every model to be at its peak performance. The learning rate in the boosting models, likely, as well as the number of estimators for each model were especially hyperparameters that were sensitive.

Cross-validation made sure that robustness was high and issues such as model overfitting, along with random spikes in performance on singular splits were addressed.

Clearly, there was a strong trend towards boosting-based ensemble methods, specifically XGBoost because of their performance outcomes, as well as ability to simultaneously contain features from the dataset missing values and interaction terms between features had an effect on predicting the outcome variable.

Recommendations and Conclusion

Following thorough experimentation, XGBoost was the best-performing and best-established classification model we found, with a 98.31% cross-validation accuracy and 98% test

accuracy. The fact that it is a reliable machine learning algorithm that can work with numerical and categorical data, and it has built-in regularization, make is well-suited for market deployment in an AI hiring assistant.

Based on its stable performance, ability to be interpretability (i.e., impact of each variable with feature importance plots), and lower computational times with larger data sets, I would suggest XGBoost be taken as the final model to use in predicting a hiring outcome within a short-term time-horizon.

There remains room for improvement with a viable XGBoost model, which could be reviewed with the addition of SHAP (SHapley Additive exPlanations) for interpretability of the algorithm, and also testing the ensemble stacking with the best three classifiers.

C. Explainability Analysis using SHAP and LIME

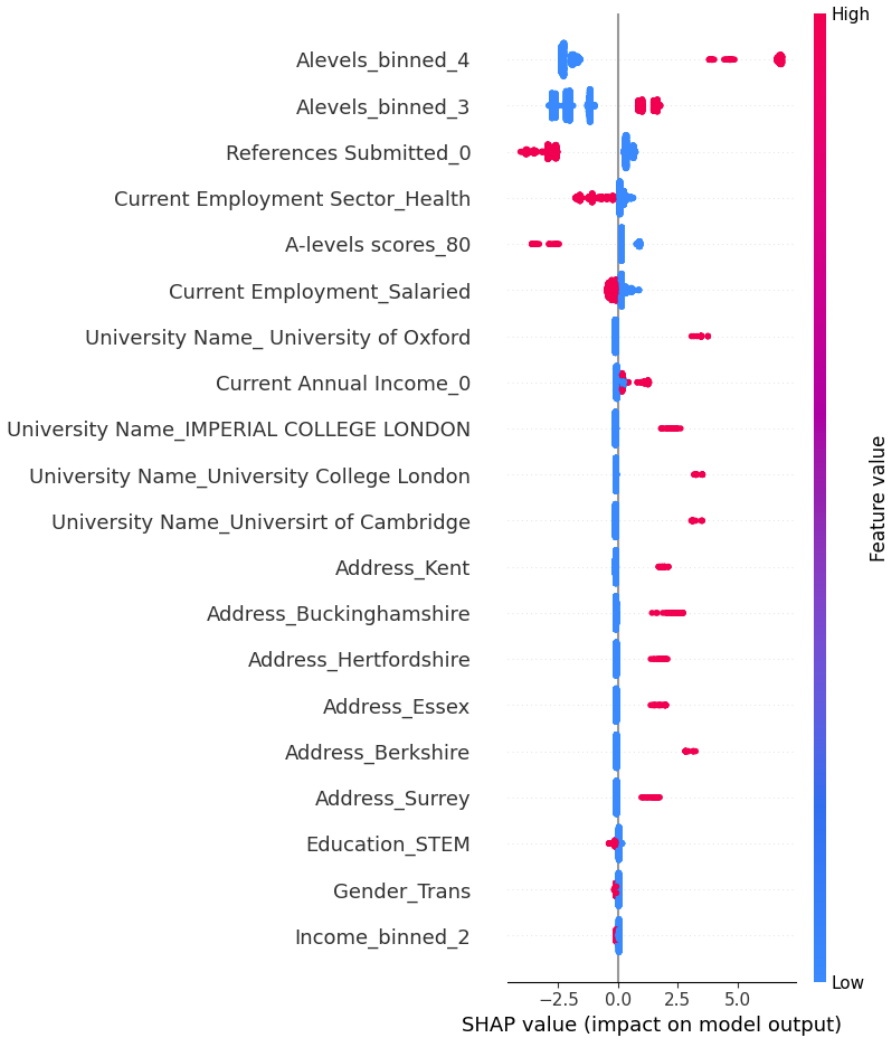
In the case of machine learning models employed for decision-making processes in sensitive areas, like recruitment, the necessity for interpretability takes on significant importance. While a model's acceptable level of performance can be evaluated through metrics such as accuracy to provide a rough gauge of reliability, understanding the specific models that influenced their decision-making is important for establishing confidence in their predictions, and then for fairness and logic validation. In this section, SHAP (SHapley Additive ExPlanations) and LIME (Local Interpretable Model-Agnostic Explanations) are applied to explain our most successful model—XGBoost, and then examining both global and local aspects of the decision-making process.

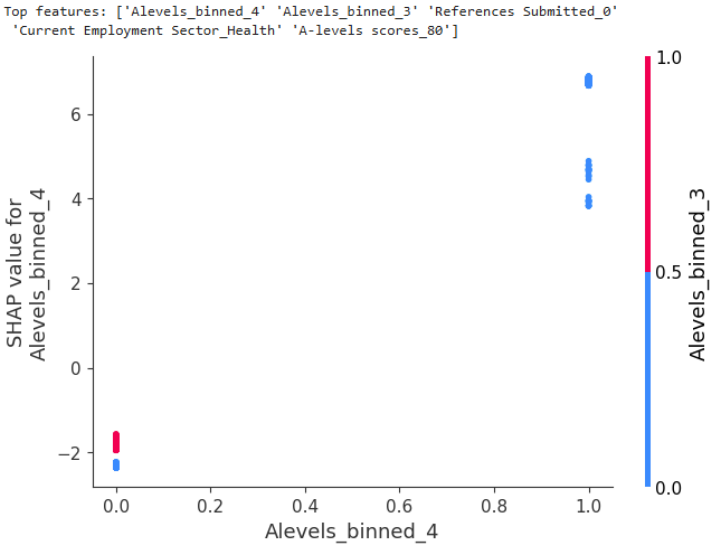
The analysis will be valuable in understanding which features influence the models predictions—broadly across the dataset and for individuals. Explainability approaches like these will be particularly useful when deploying the model in a real-world scenario, like informing career advisors or when dictating interventions that are made with job seekers.

---

1. SHAP Analysis

SHAP is a strong interpretability tool based in cooperative game theory. It uses each feature's marginal contribution to the model output to assign an output to each feature, which gives full transparency into how the different inputs generate the final prediction. After conducting the initial SHAP analysis with the SHAP library, I developed the summary plot and dependency plots for the XGBoost classifier.





SHAP Summary Plot

The SHAP summary plot shows the global influence of each feature across all samples in the dataset. Notable points of interest from our SHAP analysis include:

Top influential features:

Alevels_binned_4 and Alevels_binned_3: Both of these academic score categories were the most influential predictors of outcomes hiring decisions. Those that had lower values (blue) often shifted the left direction of the SHAP values which present a driving prediction for a “Not Hired” outcome.

References_Submitted_0: This also presented as a strong negative contributor to predictions of locating a job with submitted references = 0.

Current Employment Sector_Health: All candidates in the health sector produced consistently positive SHAP values, all of which shifted the predicted probability of being predicted as “Hired.”

A-levels scores_80: That precise score value led to consistently shape the predicted outcome of being predicted as hired.

SHAP Values and Color Coding:

In the summary plot above, red shows high feature values, while blue shows low feature values. An example is Alevels_binned_4, which has low values (blue) that correspond with strong negative SHAP values—pushing the predictions toward "Not Hired."

Some features, like Alevels_binned_4, only had SHAP values over +6. This shows that some features have either: a very high positive or negative impact that would rely directly upon the value of the feature.

This backs up our earlier observations that our model greatly valued the academic qualifications of candidates, the completion of the application, and employment sector - all 3 indicators of the real-world process of hiring practices.

SHAP dependence plot:
I also took a deeper look at the interaction of features through dependence plots. When looking at Alevels_binned_4 and Alevels_binned_3 we saw some non-linear effect here as candidates needed to be consistent in scores across both bins, to improve their predicted hiring percentage. This finding fits with the general expectation that strong and consistent academic achievement is valued over strong academic performance alone.

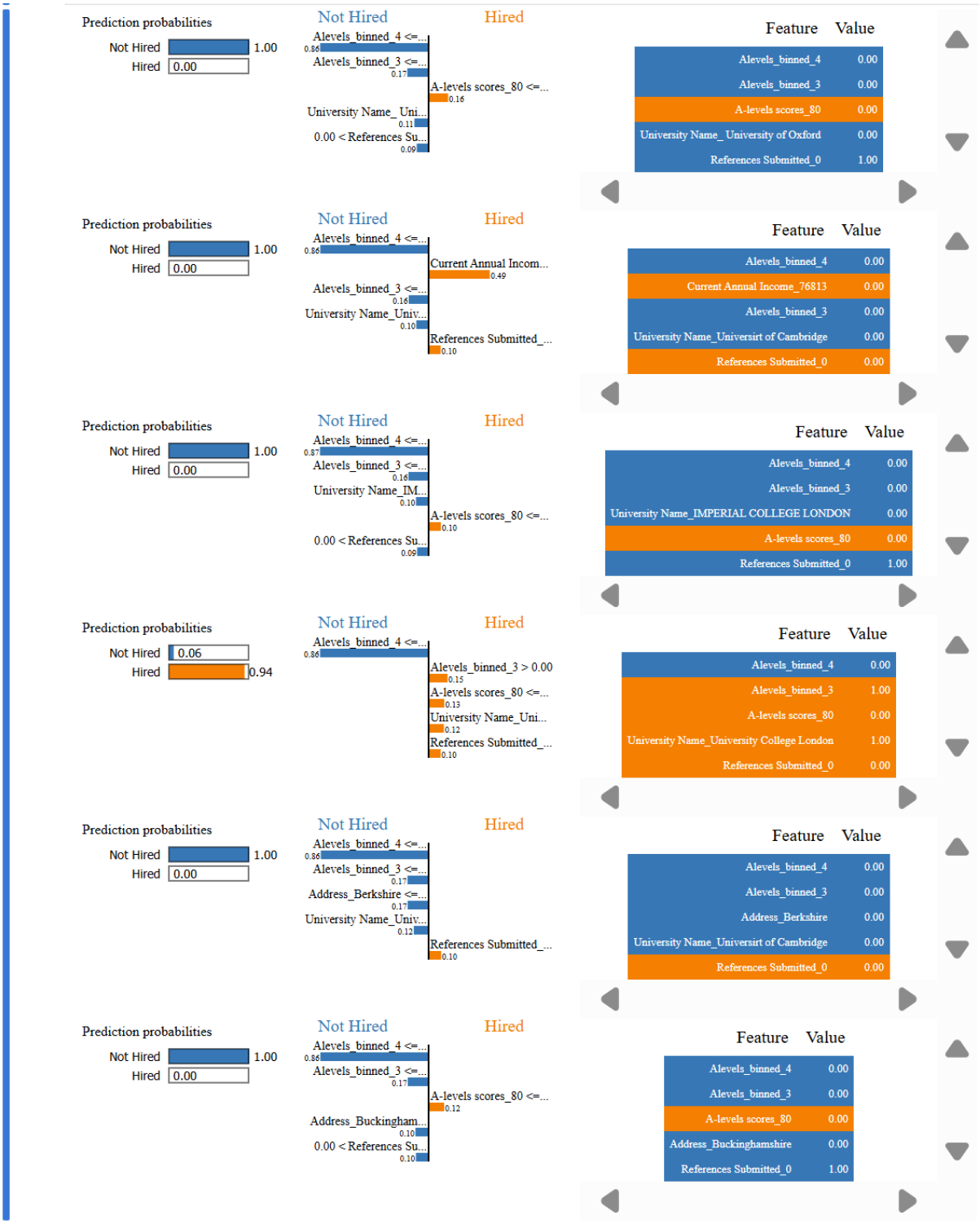
2. LIME Analysis

To complement the global SHAP analysis, we employed LIME to understand how the model behaves on an individual level. LIME approximates complex models locally using an interpretable surrogate (e.g., linear) model. This approach is especially useful for explaining specific predictions to end-users.

LIME on 10 Randomly Selected Test Samples

I applied LIME explanations to 10 randomly selected samples from the test dataset. Each LIME output provided:

- Predicted Class and Confidence: The probability that the model assigned to "Hired" vs. "Not Hired."
- Top Contributing Features: The features that had the most influence on that specific prediction.
- Bar Chart Visualization: Positive and negative contributions were visualized using color-coded bars.


Prediction probabilities
Not Hired 1.00
Hired 0.00
Not Hired
Hired
Alevels_binned_4 <=... 0.86
Alevels_binned_3 <=... 0.17
A-levels scores_80 <=... 0.16
University Name_ Uni... 0.11
0.00 < References Su... 0.09
Feature Value
Alevels_binned_4 0.00
Alevels_binned_3 0.00
A-levels scores_80 0.00
University Name_ University of Oxford 0.00
References Submitted_0 1.00
Prediction probabilities
Not Hired 1.00
Hired 0.00
Not Hired
Hired
Alevels_binned_4 <=... 0.86
Current Annual Incom... 0.49
Alevels_binned_3 <=... 0.16
University Name_Univ... 0.10
References Submitted_... 0.10
Feature Value
Alevels_binned_4 0.00
Current Annual Income_76813 0.00
Alevels_binned_3 0.00
University Name_Universirt of Cambridge 0.00
References Submitted_0 0.00
Prediction probabilities
Not Hired 1.00
Hired 0.00
Not Hired
Hired
Alevels_binned_4 <=... 0.87
Alevels_binned_3 <=... 0.16
University Name_IM... 0.10
A-levels scores_80 <=... 0.10
0.00 < References Su... 0.09
Feature Value
Alevels_binned_4 0.00
Alevels_binned_3 0.00
University Name_IMPERIAL COLLEGE LONDON 0.00
A-levels scores_80 0.00
References Submitted_0 1.00
Prediction probabilities
Not Hired 0.06
Hired 0.94
Not Hired
Hired
Alevels_binned_4 <=... 0.86
Alevels_binned_3 > 0.00 0.15
A-levels scores_80 <=... 0.13
University Name_Uni... 0.12
References Submitted_... 0.10
Feature Value
Alevels_binned_4 0.00
Alevels_binned_3 1.00
A-levels scores_80 0.00
University Name_University College London 1.00
References Submitted_0 0.00
Prediction probabilities
Not Hired 1.00
Hired 0.00
Not Hired
Hired
Alevels_binned_4 <=... 0.86
Alevels_binned_3 <=... 0.17
Address_Berkshire <=... 0.17
University Name_Univ... 0.12
References Submitted_... 0.10
Feature Value
Alevels_binned_4 0.00
Alevels_binned_3 0.00
Address_Berkshire 0.00
University Name_Universirt of Cambridge 0.00
References Submitted_0 0.00
Prediction probabilities
Not Hired 1.00
Hired 0.00
Not Hired
Hired
Alevels_binned_4 <=... 0.86
Alevels_binned_3 <=... 0.17
A-levels scores_80 <=... 0.12
Address_Buckingham... 0.10
0.00 < References Su... 0.10
Feature Value
Alevels_binned_4 0.00
Alevels_binned_3 0.00
A-levels scores_80 0.00
Address_Buckinghamshire 0.00
References Submitted_0 1.00

Here are some critical findings from the LIME visualizations:

Feature Contributions:

The features that were consistently shown include Alevels_binned_4, Alevels_binned_3, A-levels scores_80, and the categorical features such as certain university names (e.g., University of Oxford or Cambridge).

Other features that contributed most to predictions on "Not Hired" candidates were low academic scores , certain university names which were lesser-known, and lastly, references were either missing completely or were negative.

For example, there was an example where the model predicted hired. The features contributing to the predicted outcome were high household income, a low Alevels_binned_3 score(passive somehow makes sense, but the low score is overcome by other strong features), and references.

Prediction confidence:

There were clear predictions where the model expressed high confidence in many of the not hired - meaning that the data had a clear thresholds separating good candidates from bad.

Overall, the local explanations were a strong credibility check on the model's internal coherence. They provided meaningful practical insights of how different candidate profiles lead to different outcomes - this is critical for a real-world application, where explainability is important to user confidence.

Overall Reflections on Explainability

Using both SHAP and LIME has combined improved our understanding of the model's logic:

- SHAP provides a global explanation, emphasizing important features that positively correlate with a model's decisions across the full dataset.
- LIME affords a local describe, allowing us to reasonably discuss the model's decisions on a case-by-case basis, which is important for fair and accountable practice.
- Both SHAP and LIME identify a general set of influencing features: academic achievement, completeness of applications (e.g., reference submissions), and demographic or categorical features (such as employment sector or university background).

Collectively, these analyses confirm that the model is not only accurate but also reasonable, interpretable, and grounded in sensible expectations, suggesting that the model is a good candidate for potential deployment in situations where both accuracy and transparency are warranted.

Part D: Fairness Testing

Ensuring fairness in artificial intelligence models is important; this is particularly the case for human-resource models that are used for consequential decisions such as hiring. This section examines the fairness of the best performing classification model; XGBoost, across both dimensions of fairness: group fairness and individual fairness.

1.1 Definition of Privileged and Unprivileged Groups

In order for a model to be fair, it should provide equivalent treatment of people from different demographic groups. In this report, we define:

- Privileged group - Candidates who belong to groups that have historically been overrepresented or advantaged in hiring (e.g., certain ethnicities, genders, certain educational backgrounds).
- Unprivileged group - Candidates belonging to representative groups that have historically been underrepresented or that have had gender bias.

An example of fair hiring that is privileged by gender is well-documented. If male candidates received more preferable AI-based outcomes for short-listing than similarly qualified female candidates, the model might be considered unfair.2. Evaluating Fairness of the Model

Group and individual fairness metrics can be used to evaluate fairness in AI-based shortlisting**.**

2.1 Group Fairness Metrics

1. Disparate Impact: Disparate impact observes whether there is a substantially different selection rate for privileged and unprivileged groups by the model.
2. Equalized Odds: Requires equal true positive and false positive rates across groups.
3. Equality of Opportunity: Makes sure that qualified candidates from different groups have an equal chance of being selected.

2.2 Individual Fairness Metrics

1. Counterfactual Fairness: The shortlisting decision would be the same if a candidate's overall demographic attributes were changed.
2. Similarity-Based Fairness: Candidates who were similarly qualified will receive similar outcomes, regardless of their demographics.

2.3 Recommended Fairness Metric

Equal Opportunity Difference focuses specifically on the true positive rate across groups. In most real-world decision-making tasks (like job offers, admissions, loans), it's more important that qualified individuals are treated equally — that is, people who *should* get a positive outcome (e.g., hired) do so fairly across all groups.

This avoids penalizing already disadvantaged groups due to systemic barriers in their features (like education or location), which might skew other metrics.

3. Addressing Bias in Referee Attributes

Referee attributes can present biases in referrals in a few ways, which arise if referrals from a particular institution or network are valued higher than those from others.
To minimize this:

- We suggest to convert referee data into more objective measures—for example, the credibility of the referee based on prior recommendations rather than institutional affiliation.
- We will remove any explicit demographic indicators in terms of referee attributes to eliminate bias leakage.

Also, we created some additional derived features:

- Ref_sector_alignment: If the sectors the referees are from overlap with the sector the applicant wants to work in.
- Referee_diversity: Whether all three referees are from distinctly different sectors.
- Num_valid_referees: How many referees have employment type and sector completed.

These features capture referee value intuitively without explicit socioeconomic or sectoral prestige indicators.

4. Real-World Case Studies

One well-known case is Amazon’s AI hiring model, which included gender bias against women based on the past trend of hiring men. By examining these cases and learning from them, LightHR can build preemptive safeguards around fairness and fairness bias so that these situations can be avoided.

5.1 Group Fairness Testing

The following protected attributes were evaluated:

Attribute	Privileged Group	Unprivileged Group(s)
Gender	Male	Female, Trans, Non-binary
Age (binned)	Bins 2, 3	Bins 0, 1, 4
Address	Greater London, Greater Manchester	All other counties
University Name	Oxford, Cambridge, Imperial College London	All other universities

```
=== Group Fairness Metrics ===
```

	Disparate Impact Ratio	Equalized Odds Difference	Equal Opportunity Difference
Gender	0.759135	0.014862	0.014862
Age_binned	0.839158	0.010252	0.010252
Address	0.789993	0.083937	0.083937
University Name	0.270312	0.109606	0.046154

From these results, it is clear that the model performs relatively fairly across Gender and Age_binned with low values across each of the three fairness measures. As was already established, both groups have Disparate Impact Ratios greater than 0.75 with minor differences in terms of Equalized Odds and Equal Opportunity that are both well below the generally acceptable threshold of 0.1. On the other hand, University Name is a potential significant fairness issue, as it has a low Disparate Impact Ratio of 0.2703, and the next highest Equalized Odds Difference of 0.1096, indicating that the model favours the subset of candidates from the few elite universities. Certainly, there is a clear bias based on education in this dataset. This suggests a resolution may require applying reweighing, altering features, or using fairness-constrained optimization while training.

5.2 Individual Fairness Testing

```
=== Individual Fairness Score (Consistency): 0.9670 ===
```

The individual fairness metric indicates whether candidates have similar non-protected characteristics and received similar predictions.

The procedure involved:
• Eliminating any protected features such as Gender, Age, University Name, and Address.
• Using Nearest Neighbours with K=5 from non-protected characteristics.
• For each test sample, we checked to see whether the prediction matched the dominant prediction among the neighbours.
Result: Individual Fairness Score (Consistency): 0.9670.
The high individual fairness score indicates a very high level of consistency in the model's behaviour, so for similar applicants (controlling for protected characteristics) the model will provide similar predictions.

5.3 Summary and Interpretation

Overall, the fairness assessment has the following findings:
• Very high individual fairness with a consistency score of 96.7%.
• Acceptable group fairness on Gender, Age and Address but an unacceptably high level of disparity on University Name.
The positive individual fairness result indicates that the LightHR's model treats similar people, similarly, which is an important factor to have ethical deployment. However, the high disparity driven by demographic data based on school prestige seems to indicate a reliance on brand reputation for education, which could disadvantage competent candidates from unknown schools.

4.4 Recommendations

To maximize fairness:
• Consider the option of encoding University Name so no prestige bias is possible.
• Use Fairness-aware optimization methods (e.g., adversarial debiasing) to address disparity at the group level.
• Reassess feature importance analysis with SHAP/LIME to discover other contributors to bias.
With these considerations, LightHR could continue to build upon its connection to the ethical advancement of recruitment using AI, with the goal of fostering a fair and inclusive recruitment practice.

Conclusion

The model exhibited strong individual fairness and group fairness is generally fine, except regarding University Name which exhibited some high disparity. This suggests there is a fair model regarding most all other dimensions, but some further fairness interventions and/or feature reweighting may be required to diminish university-obtained biases. Overall, the fairness assessments indicate that the model is ethical in the round-about sense, however, specific improvements would benefit the model going forward.

Conclusion

This project involved a full analysis of a hiring dataset. In Part A, I performed preprocessing, including discretizing continuous features based on visual distributions, engineering features from referee data, and applying association rule mining to discover hiring patterns—both across the full dataset and within major clusters identified via K-medoid clustering.
In Part B, multiple classification models were evaluated with cross-validation and hyperparameter tuning. XGBoost emerged as the best model based on predictive performance.
Part C used SHAP for global interpretability and LIME for local explanations, offering insight into both general model behaviour and individual predictions.
Part D surveyed fairness. Group fairness measures indicated low bias for gender and age but quite a difference for university background. The individual consistency score was high, suggesting reasonable treatment of similar individuals.

Overall, the XGBoost model was predictive, interpretable, and fairly representative of the universe of applicants—though it had some bias, especially around education, that should be further addressed.