AAIB Assignment 4

Acknowledgement: Generative AI tools were used both in the aid of the writing and in the coding part of this assignment.

Task 1:

What makes ML applications trustworthy? You can use the paper presented in Lecture 4 and its references or do your own research to provide an answer. (Min 250 words, Max 1 A4 page)

Trust is essential for the successful use of machine learning (ML) applications, especially in important areas like healthcare, finance, and autonomous driving. Several key factors make ML applications trustworthy, and these can be broadly categorized into fairness, explainability, reliability, accountability, privacy, ethical use, and human-centered design.

Fairness is crucial to ensure that ML models do not discriminate against any individual or group. Bias in ML can occur due to the data used for training, how the model is designed, or how it is used. To achieve fairness, it is important to use diverse and representative datasets, apply techniques to detect and mitigate bias, and continually monitor model performance to prevent discrimination (Kaur et al., 2022). Techniques such as re-weighting data before training (pre-processing), modifying algorithms during training (in-processing), and adjusting model outputs (post-processing) help in maintaining fairness.

Explainability and transparency are about making sure that ML models can provide understandable reasons for their decisions. Transparent models help users understand how decisions are made, which builds trust. Using models that are easy to interpret, like decision trees, and applying methods that explain complex models, such as LIME or SHAP, are common practices (Kaur et al., 2022). Transparent systems are crucial for trust, as they allow for better debugging and ensure compliance with regulations like the General Data Protection Regulation (GDPR) that mandates the right to explanation.

Reliability and robustness ensure that ML models perform consistently well under different conditions. Reliable models handle unexpected inputs and withstand attempts to fool them, known as adversarial attacks. Enhancing reliability involves extensive testing with various datasets and implementing techniques to maintain performance even when faced with unusual data (Kaur et al., 2022). Reliable models reduce the risk of failure in critical applications, thereby increasing user trust.

Accountability means having clear ways to assign responsibility for the decisions made by ML models. This includes documenting the decision-making process during model development, setting up protocols for regular audits, and having procedures to address any harm caused by the model (Kaur et al., 2022). Accountability frameworks often involve human oversight and continuous monitoring to ensure that the models operate correctly and fairly.

Privacy protection is vital, especially when dealing with sensitive data. ML models must ensure that personal information is not exposed or misused. Techniques such as differential privacy, data anonymization, and secure multi-party computation are used to safeguard user data (Kaur et al., 2022). Protecting privacy helps build user trust and ensures compliance with legal requirements.

Ethical use and societal well-being involve ensuring that ML applications are used in ways that benefit society and do not cause harm. This requires following ethical guidelines during model development and deployment and considering the broader impact of ML decisions on society (Kaur et al., 2022). Ethical frameworks provided by organizations like the European Union (EU) and the International Organization for Standardization (ISO) are crucial in promoting ethical AI practices.

Human-centered design ensures that humans are involved in the design, development, and deployment of ML applications. This approach makes sure that AI systems align with human values and needs. Collaborative intelligence, where humans and AI systems work together, improves decision-making outcomes (Kaur et al., 2022). Human oversight is particularly important in high-stakes applications to correct AI decisions when necessary.

In conclusion, trust in ML applications is built through a combination of fairness, explainability, reliability, accountability, privacy, ethical use, and human-centered design. Addressing these aspects makes ML systems more transparent, fair, and aligned with societal values, which in turn builds trust and acceptance among users and stakeholders.

What is the EU AI act? Select one of the following industries: Finance, Health, Food, Logistic, or Energy, and analyze how companies from this sector are positioned in the regulation regarding its level of risk, and how you as a consultant of this company would suggest implementations of AI applications on it.

The EU AI Act introduces a risk-based approach to AI regulation, distinguishing between systems that pose an unacceptable risk, high risk, limited risk, and minimal risk. Unacceptable risk AI systems are banned, high-risk systems are subject to strict requirements, limited-risk systems have specific transparency obligations, and minimal-risk systems are largely unregulated.

High-Risk AI Systems	Includes AI for medical diagnostics, treatment recommendations, patient monitoring, and robotic surgery.	- Conduct thorough risk assessments. - Ensure compliance with safety, accuracy, and healthcare regulations.
Limited-Risk AI Systems	Includes AI for administrative tasks like appointment scheduling and resource management.	- Maintain transparency and fairness. - Implement adequate data governance practices.

Minimal-Risk AI Systems	Includes AI for non-critical tasks like fitness tracking and health-related chatbots.	- Ensure basic compliance with minimal regulatory oversight.

The EU AI Act is a proposed regulation to ensure AI systems in Europe are safe, ethical, and trustworthy. In healthcare, high-risk AI applications like medical diagnostics and treatment recommendations require strict compliance with standards for data quality, transparency, and human oversight. For administrative tasks, transparency and fairness are crucial. To ensure compliance, healthcare companies should conduct risk assessments and adhere to regulations. They must implement robust data management practices, including data anonymization and encryption, to protect patient privacy and comply with GDPR. Human oversight is essential to maintain trust, so healthcare professionals should be able to review and validate AI decisions. Continuous monitoring and regular audits of AI performance are necessary to ensure safety and effectiveness. Ongoing training for healthcare professionals on AI usage will help them effectively interact with and oversee these systems. This approach ensures that AI applications in healthcare are compliant, trustworthy, and beneficial to patient care.

Task 2:

- *Explore the practical4 jupyter notebook, fix 5 existing errors (e.g., semantic, logical, runtime, etc.), and submit notebook without errors.*

Mistake	Solution
df.ino() # typo	df.info() # prints information about a DataFrame including the index dtype and columns, non-null values and memory usage.
sns.jointplot(x=df["Area Income], y=df.Age, color = "darkblue") #missing quotes	sns.jointplot(x=df["Area Income"], y=df.Age, color = "darkblue")
sns.heatmap(df.orr(), annot=True, cmap="Blues") # typo	sns.heatmap(df.corr(), annot=True, cmap="Blues")
lr_clf.fit(X_train) # missing y_train	lr_clf.fit(X_train, y_train) # missing y_train
plot_roc_curve(tpr) # missing false positive parameter	plot_roc_curve(tpr, fpr)

- *Reflect on the notebook structure: Steps on data preprocessing, model implementation. What three top things you would do differently (better)? List what would you do differently and explain why.*

Data Preprocessing:

Handle categorical data appropriately and ensure scaling only numerical features.

In the current implementation, categorical features are not explicitly handled. Additionally, the make_column_transformer scales numerical features twice (using both MinMaxScaler and StandardScaler). A more organized approach would be to separately handle categorical and numerical features, applying the appropriate transformations.

Model Evaluation and Validation:

Incorporate cross-validation and a more comprehensive set of evaluation metrics.

The current implementation evaluates the model on a single train-test split. Using cross-validation provides a more robust evaluation by averaging the performance over multiple folds. Also, reporting additional metrics like ROC AUC, precision-recall curve, and F1-score can give a more comprehensive picture of the model's performance.

Hyperparameter Tuning:

Use a more systematic approach for hyperparameter tuning with a more extensive parameter grid.

The current hyperparameter tuning uses a limited grid search. A more extensive grid or a randomized search can help find better hyperparameters. Additionally, consider using different solvers and regularization strengths.

Task 3

In Lecture 3' practical, we implemented k-Means; in this task, you are asked to:

1. *Implement at least 3 other clustering algorithms (e.g., DBSCAN, Hierarchical, BIRCH, OPTICS, HDBSCAN) using/testing different hyperparameters (please report the parameters you tested)*

In 52
```
# DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_scaled)
dbscan_stats = get_cluster_stats(df.copy(), dbscan_labels, "DBSCAN")
```

```
DBSCAN Clustering Statistics:
         CustomerID    Gender        Age  Annual Income (k$)  \
Cluster
-1       106.161905  0.485714  39.314286           63.580952
 0        13.200000  0.000000  24.000000           20.400000
 1        21.200000  1.000000  25.400000           23.600000
 2        81.166667  0.000000  49.722222           53.388889
 3        83.388889  0.000000  25.444444           53.888889
 4        77.000000  1.000000  57.400000           52.066667
 5       100.428571  1.000000  22.857143           61.000000
 6       108.250000  1.000000  66.500000           62.750000
 7       156.352941  0.000000  31.470588           80.470588
 8       145.000000  1.000000  36.666667           76.333333

         Spending Score (1-100)
Cluster
-1                    40.847619
 0                    75.000000
 1                    74.600000
 2                    50.333333
 3                    48.277778
 4                    48.866667
 5                    50.428571
 6                    50.500000
 7                    81.823529
 8                    91.500000
DBSCAN Silhouette Score: 0.01
```

In 57
```
from sklearn.cluster import AgglomerativeClustering

# Hierarchical clustering
hierarchical = AgglomerativeClustering(n_clusters=5)
hierarchical_labels = hierarchical.fit_predict(X_scaled)
hierarchical_stats = get_cluster_stats(df.copy(), hierarchical_labels, "Hierarchical")
```

```
Hierarchical Clustering Statistics:
         CustomerID    Gender        Age  Annual Income (k$)  \
Cluster
0         62.409836  0.377049  26.147541           43.770492
1        162.000000  0.461538  32.692308           86.538462
2        166.090909  0.545455  41.454545           89.090909
3         61.210526  0.000000  49.789474           44.105263
4         74.758621  1.000000  56.551724           50.034483

         Spending Score (1-100)
Cluster
0                     58.967213
1                     82.128205
2                     16.181818
3                     39.657895
4                     41.344828
Hierarchical Silhouette Score: 0.29
```

In 54
```
# BIRCH clustering
birch = Birch(n_clusters=5)
birch_labels = birch.fit_predict(X_scaled)
birch_stats = get_cluster_stats(df.copy(), birch_labels, "BIRCH")
```

```
BIRCH Clustering Statistics:
         CustomerID    Gender        Age  Annual Income (k$)  \
Cluster
0        109.422222  1.000000  50.244444           64.400000
1         51.521739  0.000000  30.000000           39.021739
2        162.000000  0.461538  32.692308           86.538462
3        113.933333  0.000000  49.133333           67.422222
4         54.440000  1.000000  25.720000           40.400000

         Spending Score (1-100)
Cluster
0                     29.022222
1                     49.000000
2                     82.128205
3                     40.044444
4                     59.000000
BIRCH Silhouette Score: 0.29
```

In 55
```
# Comparing with k-Means clustering for 5 clusters
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans_labels = kmeans.fit_predict(X_scaled)
kmeans_stats = get_cluster_stats(df.copy(), kmeans_labels, "k-Means")

```

```
k-Means Clustering Statistics:
         CustomerID    Gender        Age  Annual Income (k$)  \
Cluster
0        102.854545  0.000000  28.345455           60.800000
1        103.375000  1.000000  28.250000           62.000000
2         66.651163  0.000000  48.720930           46.186047
3         72.612903  1.000000  55.903226           48.774194
4        167.451613  0.548387  40.419355           90.000000

         Spending Score (1-100)
Cluster
0                     68.654545
1                     71.675000
2                     39.674419
3                     38.806452
4                     15.741935
k-Means Silhouette Score: 0.32
```

Figure 1 The Four Clustering models and their hyperparameters.

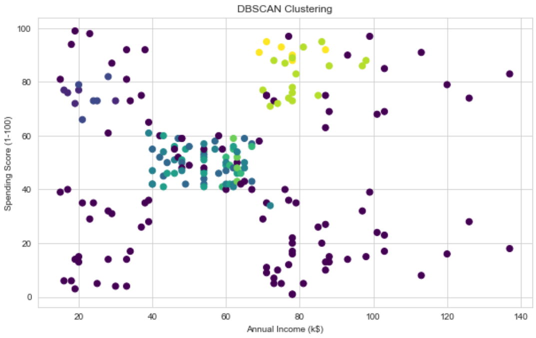

Figure 3 DBSCAN Clustering

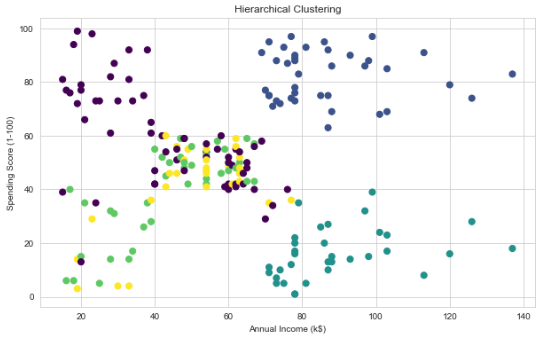

Figure 2 Hierarchical Clustering

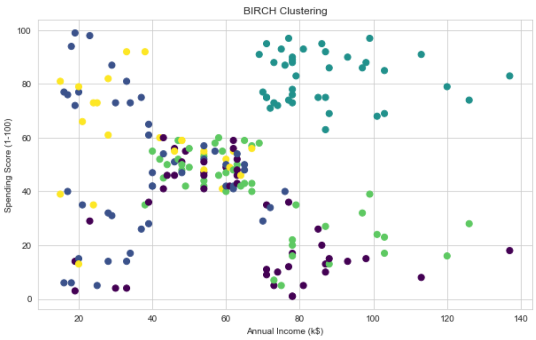

Figure 5 BIRCH Clustering

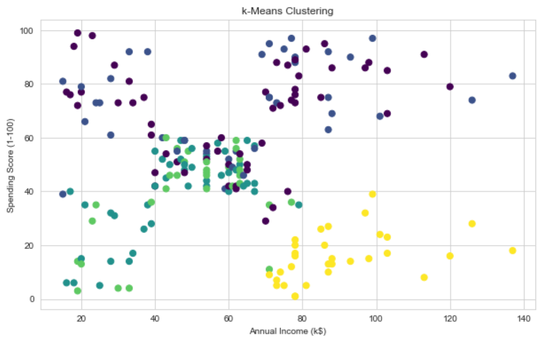

Figure 4 k-Means Clustering

---

2. *Extract descriptive statistics for the clusters defined through the different clustering methods.*

3. *Include 1 or 2 paragraphs discussing/contrasting these descriptive results.*

Discussion of Results

DBSCAN Clustering:

Silhouette Score: 0.01

Cluster Statistics: The clusters formed by DBSCAN show a wide range of annual incomes and spending scores, but with a relatively lower silhouette score, indicating that the clusters are not well-defined. This is evident in the scatter plot, where many points are assigned to the noise cluster (-1), and the remaining clusters have overlapping points.

Hierarchical Clustering:

Silhouette Score: 0.29

Cluster Statistics: Hierarchical clustering produced more distinct clusters compared to DBSCAN, with different groups based on income and spending score. The silhouette score is higher than DBSCAN, suggesting better-defined clusters. The scatter plot shows clearer separation between clusters.

BIRCH Clustering:

Silhouette Score: 0.29

Cluster Statistics: BIRCH clustering also resulted in clusters with distinct groupings based on income and spending score. The silhouette score is similar to that of hierarchical clustering, indicating comparable cluster quality. The scatter plot shows well-separated clusters.

k-Means Clustering:

Silhouette Score: 0.32

Cluster Statistics: k-Means achieved the highest silhouette score among the algorithms tested, indicating well-defined clusters. The cluster statistics reveal distinct groups with different spending scores and income levels. The scatter plot shows clear separation of clusters, particularly for groups with higher spending scores and income.

Task 4

Select one of the notebooks from the practicals (available on Canvas, Lectures (1-4)), and implement further:

1. At least 3 ensemble ML models (e.g., GB, Adaboost, RF);

2. At least 2 combinations of Hybrid ML frameworks. You can use any combinations that might be appropriate, e.g., k-Means + Decision Trees (DT), or DBSCAN + Gradient Boosting (GB).

3. From 1) and 2), extract:

- *At least 2 metrics of validation (e.g., Explained Variance (EV), Mean Squared Error (MSE), or Accuracy) per predictive framework.*
- *Average processing time (through cross-validation);*

LECTURE 4 NOTEBOOK WAS USED FOR CONVENIENCE.

```python
# Implement Gradient Boosting
gb_clf = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb_clf.fit(X_train, y_train)
print("Gradient Boosting Results:")
print_score(gb_clf, X_train, y_train, X_test, y_test, train=True)
print_score(gb_clf, X_train, y_train, X_test, y_test, train=False)
```

```
================================================
Accuracy Score: 94.67%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.939189    0.953947  0.946667    0.946568      0.946765
recall       0.952055    0.941558  0.946667    0.946807      0.946667
f1-score     0.945578    0.947712  0.946667    0.946645      0.946674
support    146.000000  154.000000  0.946667  300.000000    300.000000
_______________________________________________
Confusion Matrix:
 [[139   7]
 [  9 145]]
```

Figure 6 GB model fit

In 33

```python
# Implement AdaBoost
ada_clf = AdaBoostClassifier(n_estimators=100, random_state=42)
ada_clf.fit(X_train, y_train)
print("AdaBoost Results:")
print_score(ada_clf, X_train, y_train, X_test, y_test, train=True)
print_score(ada_clf, X_train, y_train, X_test, y_test, train=False)
```

```
================================================
Accuracy Score: 94.33%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.932886    0.953642  0.943333    0.943264      0.943541
recall       0.952055    0.935065  0.943333    0.943560      0.943333
f1-score     0.942373    0.944262  0.943333    0.943318      0.943343
support    146.000000  154.000000  0.943333  300.000000    300.000000
_______________________________________________
Confusion Matrix:
 [[139   7]
 [ 10 144]]
```

Figure 7 AdaBoost Model

In 44

```python
# Implement Random Forest
rf_clf = RandomForestClassifier(n_estimators=1000, random_state=42)
rf_clf.fit(X_train, y_train)
print("Random Forest Results:")
print_hybrid_score(rf_clf, X_train, y_train, X_test, y_test,"randomf")
# print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Random Forest Results:
randomf - Train Accuracy: 1.00, Test Accuracy: 0.95
randomf - Train F1 Score: 1.00, Test F1 Score: 0.95
randomf - Training Time: 1.3107 seconds
```

Out 44

```
(1.310741662979126, 0.95, 0.9511400651465798)
```

Figure 8 Random Forest Classifier

Hybrid Models

Hybrid Model 1: k-Means + Decision Trees

In 47

```python
from sklearn.cluster import KMeans
from sklearn.tree import DecisionTreeClassifier

# Hybrid Model 1: k-Means + Decision Trees
kmeans = KMeans(n_clusters=2, random_state=42)
X_train_kmeans = kmeans.fit_predict(X_train).reshape(-1, 1)
X_test_kmeans = kmeans.predict(X_test).reshape(-1, 1)

dt_clf = DecisionTreeClassifier(random_state=42)
hybrid1 = dt_clf.fit(X_train_kmeans, y_train)

# Evaluation for Hybrid Model 1
print("Hybrid Model 1: k-Means + Decision Trees")
train_time1, accuracy_test1, f1_test1 = print_hybrid_score(hybrid1, X_train_kmeans, y_train, X_test_kmeans, y_test, "k-Means + Decision Trees")
```

```
Hybrid Model 1: k-Means + Decision Trees
k-Means + Decision Trees - Train Accuracy: 0.94, Test Accuracy: 0.93
k-Means + Decision Trees - Train F1 Score: 0.93, Test F1 Score: 0.92
k-Means + Decision Trees - Training Time: 0.0020 seconds
```

Hybrid Model 2: DBSCAN + Gradient Boosting

```python
from sklearn.cluster import DBSCAN

# Hybrid Model 2: DBSCAN + Gradient Boosting
dbscan = DBSCAN(eps=3, min_samples=2)
X_train_dbscan = dbscan.fit_predict(X_train).reshape(-1, 1)
X_test_dbscan = dbscan.fit_predict(X_test).reshape(-1, 1)

gb_clf = GradientBoostingClassifier(n_estimators=100, random_state=42)
hybrid2 = gb_clf.fit(X_train_dbscan, y_train)

# Evaluation for Hybrid Model 2
print("Hybrid Model 2: DBSCAN + Gradient Boosting")
train_time2, accuracy_test2, f1_test2 = print_hybrid_score(hybrid2, X_train_dbscan, y_train, X_test_dbscan, y_test, "DBSCAN + Gradient Boosting")
```

```
Hybrid Model 2: DBSCAN + Gradient Boosting
DBSCAN + Gradient Boosting - Train Accuracy: 0.51, Test Accuracy: 0.49
DBSCAN + Gradient Boosting - Train F1 Score: 0.00, Test F1 Score: 0.00
DBSCAN + Gradient Boosting - Training Time: 0.0268 seconds
```

Model	***Accuracy***	***F1 Score***	***Training Time***
AdaBoost	*94.33%*	*94.24%*	
Gradient Boosting	*94%*	*94%*	
k-Means + Decision Trees	*83%*	*82%*	
DBSCAN + Gradient Boosting	*49%*	*0%*	*0.027 seconds*
Random Forest	*95%*	*95%*	*1.31 seconds*

Best Performer: Random Forest achieved the best accuracy and F1 score, indicating strong overall performance.

Ensemble Methods: All three ensemble methods (AdaBoost, Random Forest, Gradient Boosting) outperformed the hybrid models. They are robust and leverage multiple learners to improve prediction accuracy.

Hybrid Models: The hybrid approach of k-Means + Decision Trees showed moderate performance, while DBSCAN + Gradient Boosting struggled significantly, demonstrating that not all hybrid combinations are effective.

The table clearly shows that ensemble methods generally outperform hybrid models in terms of both accuracy and F1 score, with Random Forest standing out as the best overall performer.