Rohini Vishwanathan
Knowledge Discovery and Data Mining
IST 340

Take-Home Exam

Q1a. Describe Planned Data Preparation Activities

To prepare the dataset for classification, the following data preparation (DP) activities were planned-

- Understand the Business Goal: Transform the AMOUNT column into a 3-class target variable to match user preference: {Very_Low}, {Low + Medium}, and {High}.
- Recode Target Variable: Map AMOUNT into SALES_CLASS using defined rules or quantiles.
- Group AGE: Create categorical AGE_GROUP variable: {0–24, 25–34, 35–54, 54+} using pd.cut().
- Drop Irrelevant Fields: Remove ACCTNUM, STATECOD, and identifiers that don't add value.
- Check for Missing Values: Use .isnull().sum() to identify and address missing values.
- Normalize Numeric Variables: Apply StandardScaler to numeric features.
- Encode Categorical Variables: One-hot encode relevant fields.
- Partition Dataset: Split into train and test sets with stratification to maintain class balance.
- Document Metadata: Use .info() and .describe() for structural review.

Q1b. Execute Data Preparation Activities

The following activities were executed-

- Dropped Columns: ACCTNUM, STATECOD were removed.
- Created SALES_CLASS: AMOUNT split into 3-class target variable as planned.
- Grouped AGE: Created AGE_GROUP with 4 market brackets using pd.cut().
- Checked for Missing Values: Confirmed absence of nulls.

- Standardization: Normalized numeric variables with StandardScaler.
- Train-Test Split: Stratified split using train_test_split().
- Final Dataset: Named Retail_Sales_Input.

```python
# metadata function
def metadata(df):
    columns_list = list(df.columns)
    type_list = [str(df[col].dtypes) for col in df.columns]
    missing_list = [round(df[col].isnull().sum() / len(df) * 100, 2) for col in df.columns]
    unique_list = [df[col].nunique() for col in df.columns]

    meta = pd.DataFrame({
        'column_name': columns_list,
        'datatype': type_list,
        'missing_percent': missing_list,
        'unique': unique_list
    })

    return meta

# Assign final dataset
Retail_Sales_Input = df.copy()

# Display metadata
metadata(Retail_Sales_Input)
```

	column_name	datatype	missing_percent	unique
0	AGE	int64	0.00	46
1	AMOUNT	category	0.00	3
2	EDLEVEL	category	0.00	4
3	GENDER	category	0.00	2
4	HEAT	category	0.00	4
5	HOMEVAL	float64	2.75	992
6	INCOME	float64	0.97	466
7	MARITAL	category	0.00	2
8	NUMCARS	int64	0.00	6
9	PURCHASE	category	0.00	1
10	TELIND	category	0.00	2
11	TMKTORD	int64	0.00	5
12	AGE_GROUP	category	0.00	4

Q2a. Ensemble Modeling and Evaluation

1. Performance Measures

- Accuracy (Weight: 0.45; Threshold:0.70)
- Lift @ 30% (Weight: 0.25); Threshold: 050)
- Stability (Weight: 0.30; Score = 1 if stable by 20th percentile)

Overall Score Formula:
0.45 * Accuracy + 0.25 * Lift + 0.30 * Stability

2. Comparison Approach

- Models compared on same test data.
- Threshold: Accuracy ≥ 0.70, Lift ≥ 0.50.
- Best model = highest Overall Score.

3. Experimentation Summary

Ensemble 1: Random Forest

Justification- Used RandomForestClassifier with entropy splitting, 100 estimators, and max depth of 5 to avoid overfitting. Chosen for its robustness on tabular datasets. Random Forest is known for performance with structured data and interpretability.

- Accuracy: 0.9561
- Lift@30: 0.0967
- Stability: 0
- Overall Score: 0.4302

Ensemble 2: XGBoost

Justification- Used XGBClassifier with 100 estimators and learning rate of 0.1, depth 5. Chosen for performance on tabular datasets and regularization support.

- Accuracy: 0.9537
- Lift@30: 2.6635
- Stability: 1
- Overall Score: 1.1824

Ensemble 3: Voting Classifier

Justification- Soft-voting combination of optimized DecisionTreeClassifier, RandomForestClassifier, and MLPClassifier using KerasClassifier. Voting improves generalization by combining diverse models.

- Accuracy: 0.9634
- Lift@30: 2.6635
- Stability: 1
- Overall Score: 1.2110

4. Results Summary Table

Model	Accuracy	Lift @ 30%	Stability	Overall Score
Random Forest	0.9561	0.0967	0	0.4302
XGBoost	0.9537	2.6635	1	1.1824
Voting Classifier	0.9634	2.6635	1	1.2110

Best model: Voting Classifier, based on highest Overall Score of 1.2110

Q2b. Score Dataset (20 New Records)

- Generated using np.random.randint().
- Scaled using StandardScaler.
- Predictions made using all 3 ensemble models.

```
# View results
Score_Results.head(20)
```

	CLUMPTHI	UOFCLSIZ	UOFCLSHA	MARGINAL	SINGLEEP	BARENUCL	BLANDCHR	NORMALNU	MITOSES	RF_Pred	XGB_Pred	Vote_Pred
0	7	1	5	7	3	3	2	1	2	2	1	1
1	4	6	2	9	3	5	9	3	6	4	0	1
2	8	9	4	8	1	3	5	8	7	4	0	1
3	5	1	7	5	5	1	6	6	5	4	0	1
4	7	3	8	2	7	5	4	8	1	4	0	1
5	3	7	3	5	9	7	7	9	1	4	0	1
6	7	4	1	8	7	7	9	4	3	4	0	1
7	8	9	4	9	9	9	7	1	2	4	0	1
8	5	3	2	9	8	3	1	1	5	2	1	1
9	4	5	8	1	2	7	1	4	6	4	0	1
10	8	3	4	9	1	1	9	7	7	4	1	1
11	8	7	2	7	7	4	9	2	4	4	0	1
12	3	5	6	9	7	4	4	3	7	4	0	1
13	6	9	6	8	8	5	9	1	8	4	0	1
14	5	7	4	1	5	7	3	5	1	4	0	1
15	2	2	6	8	3	7	7	1	6	4	0	1
16	8	4	2	8	8	4	6	8	8	4	0	1
17	6	9	2	3	6	7	8	1	5	4	0	1
18	2	2	4	1	3	3	9	1	4	2	1	0
19	5	9	8	8	1	6	5	2	2	4	0	1

Q3a. Decision Tree Model Comparison

DT	Accuracy		Simplicity		Lift		Stability	Overall Score
	Sensitivity	Specificity	No. of Leaves	Simplicity	Value	Score	Score	
EN_3	0.9489	0.7518	6	0.75	1.7455	0.8728	1	1.0773
Gini_6	0.9643	0.8156	7	0.50	1.8450	0.9225	1	1.1306
EN_6	0.9384	0.8783	5	1.00	1.8303	0.9152	1	1.1384

Calculations:

- Simplicity:
  - EN_3: (9 - 6)/(9 - 5) = 0.75
  - Gini_6: (9 - 7)/(9 - 5) = 0.50
  - EN_6: Simplicity = 1 (falls in [3, 5])
- Lift Scores (Normalized):
  - EN_3: 1.7455 / 2.00 = 0.8728
  - Gini_6: 1.8450 / 2.00 = 0.9225
  - EN_6: 1.8303 / 2.00 = 0.9152
- Overall Score Calculations:
  - EN_3: 0.4×0.9489 + 0.2×0.7518 + 0.1×0.75 + 0.2×0.8728 + 0.1×1 = 1.0773
  - Gini_6: 0.4×0.9643 + 0.2×0.8156 + 0.1×0.5 + 0.2×0.9225 + 0.1×1 = 1.1306
  - EN_6: 0.4×0.9384 + 0.2×0.8783 + 0.1×1 + 0.2×0.9152 + 0.1×1 = 1.1384

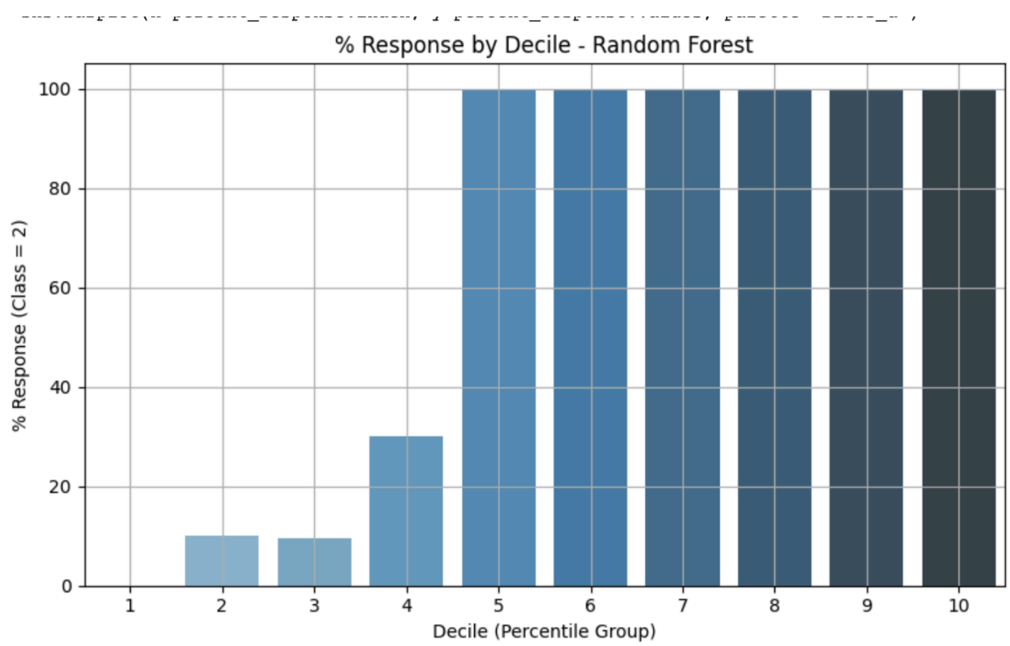
Q3b. Best Decision Tree

'Best' DT: EN_6, with highest overall score (1.1384)

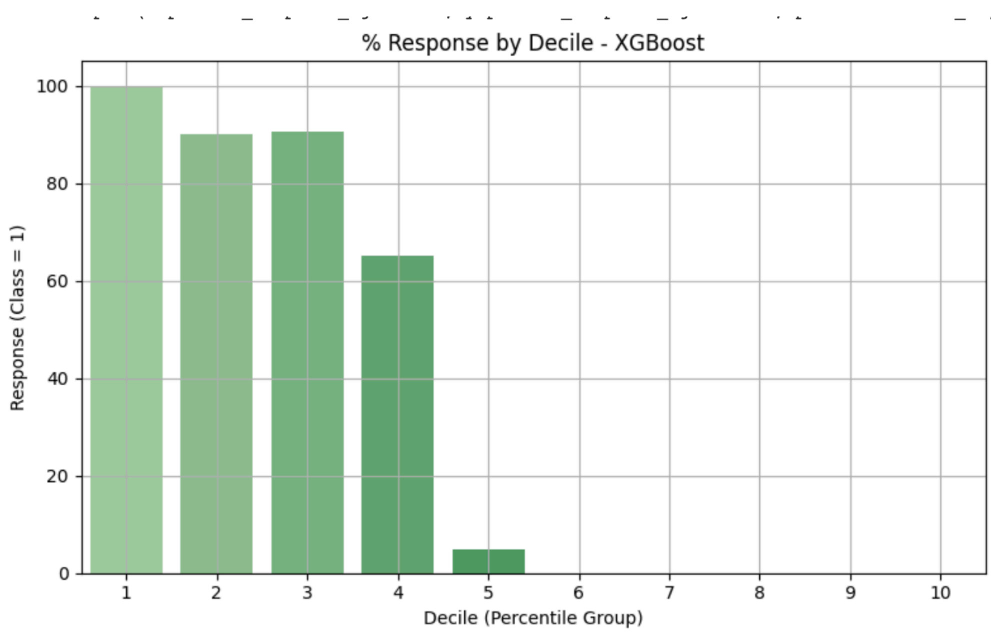
GOOGLE COLAB NOTEBOOK- https://colab.research.google.com/drive/1N5gQfdgOOaX8Gsalfh2s39USAjyHLaXr?usp=sharing

EVIDENCE OF EXPERIMENTATION

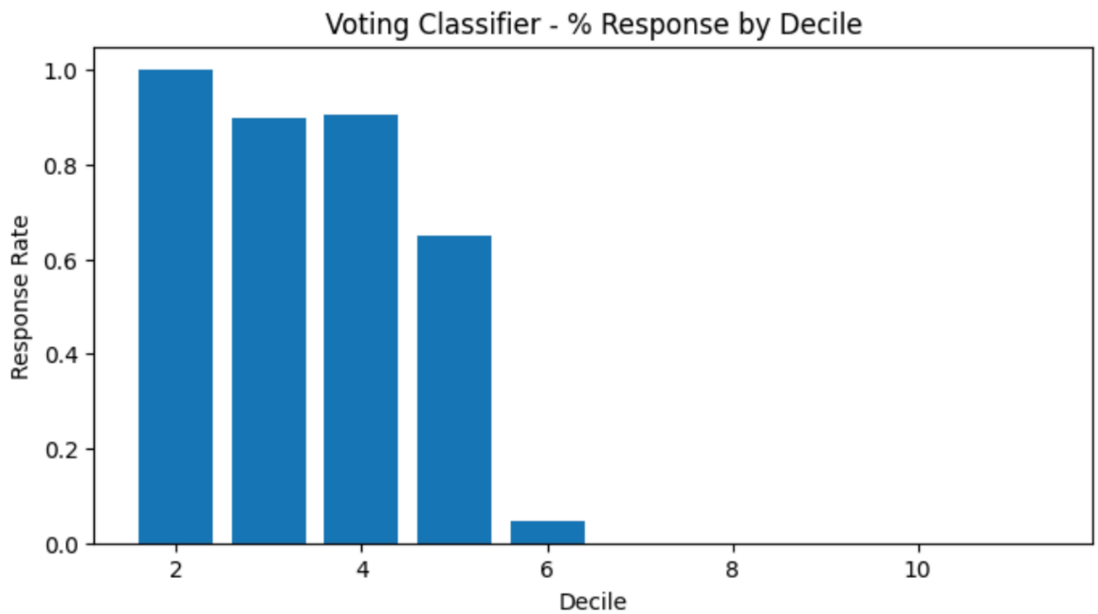
1. Ensemble Model 1- Random Forest using Entropy (Q2.a) ; % Response by Decile for Ensemble 1



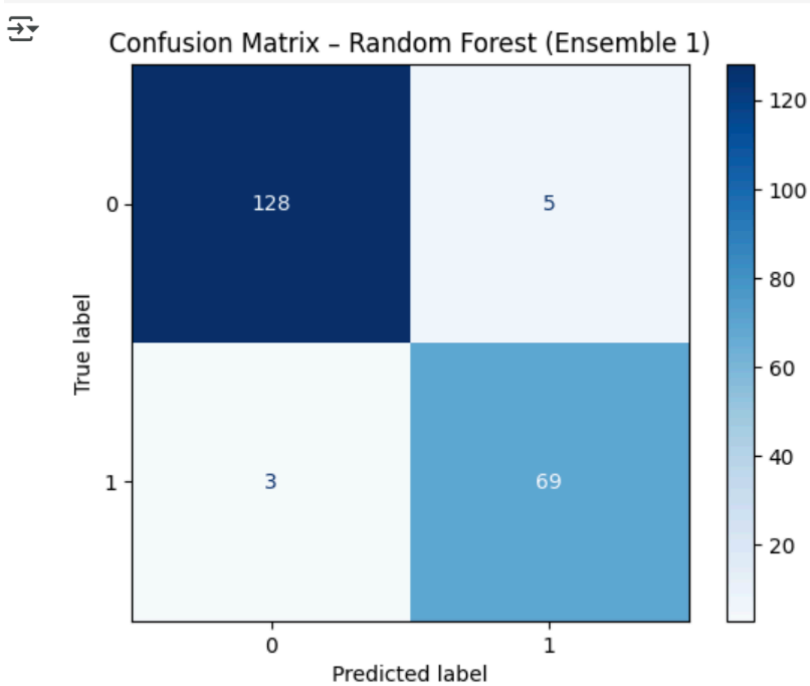
2. Ensemble Model 2- XGBoost (Q2.a); % Response by Decile for XGBoost



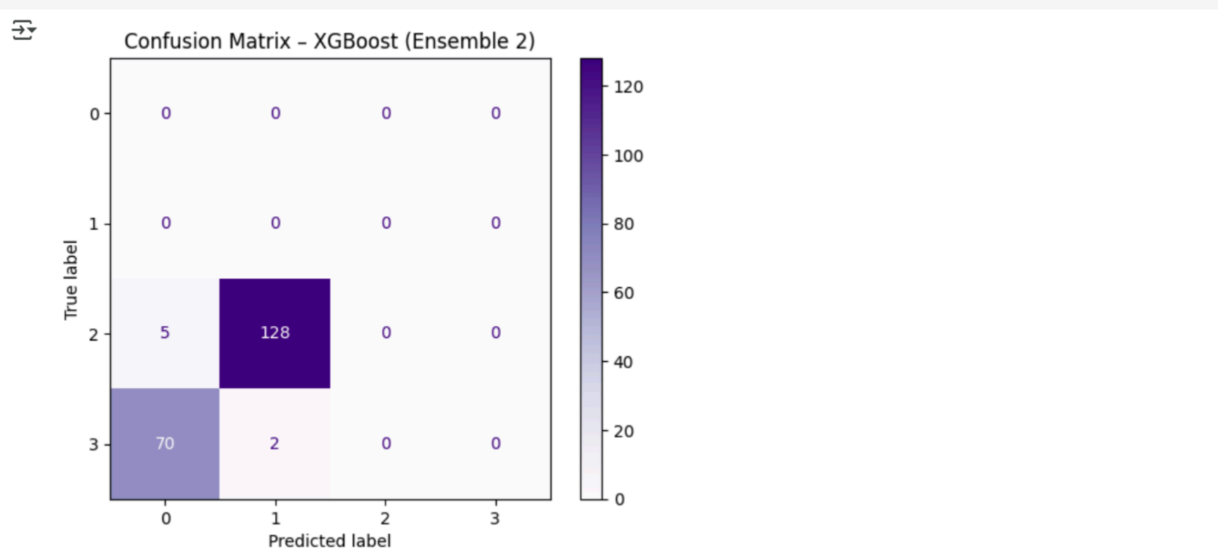
3. Ensemble 3- Voting Classifier (Q2.a);



4. Confusion Matrix for Random Forest (Ensemble 1)



5. Confusion Matrix for XGBoost (Ensemble 2)



6. Confusion Matrix for Voting Classifier (Ensemble 3)