Attribution statement **Team name:** Generalizers

Our team, whose signatures appear below, completed this project as a group effort. By our signatures, we indicate that we agree that each of us has made the following contributions.

Member 1 (enter your contributions here and then both print and sign your name)

- Led the data preprocessing phase.
- Built data transformation pipelines and used DBSCAN to detect and remove outliers.
- Worked on feature selection by researching different techniques and applying methods like Random Forest feature selection and hierarchical clustering.
- Responsible for managing the model submissions for testing on the held-out dataset.
- Reviewed results.
- Contributed to writing the final report.

Asafe Brandao.

Member 2 (enter your contributions here and then both print and sign your name)
Focused on improving the model.

- Performed hyperparameter tuning on the base learners and created visualizations to better understand the data.
- Handled final model evaluations and calculated 95% confidence intervals for the results.
- Tried different preprocessing steps for better-performing models.
- Researched and helped select evaluation metrics used throughout the project.
- Reviewed results.
- Contributed to writing the final report.

Jason Shan.

Employee **Satisfaction** (Question 61) - Performance on Our **Test Set**

Date	Attempt By	Model and Preprocessing	Balanced Accuracy	Weighted Recall	Weighted F-1	Recall [0]
05/11/25	Team	Stacked Ensemble (Meta: LogReg; Base: tuned SVC, RF, LogReg) w/ Preprocessed RF-Selected Features	0.4374	0.4299	0.4619	0.5255
05/11/25	Team	Voting Ensemble (tuned SVC, RF, LogReg) w/ Preprocessed RF-Selected Features	0.4151	0.5784	0.5736	0.2044
05/13/25	Team	Stacked Ensemble (Meta: LogReg; Base: tuned SVC, RF, LogReg) w/ Preprocessed RF-Selected Features	0.4388	0.4305	0.4589	0.5255
05/13/25	Team	Voting Ensemble (tuned SVC, RF, LogReg) w/ Preprocessed RF-Selected Features	0.4160	0.5797	0.5755	0.2044

Table 2-1: Employee Satisfaction models' performance on the test set.

Employee **Fulfillment** (Question 62) - Performance on Our **Test Set**

Date	Attempt By	Model and Preprocessing	Balanced Accuracy	Weighted Recall	Weighted F-1	Recall [0]
05/11/25	Team	Stacked Ensemble (Meta: LogReg; Base: tuned SVC, RF, LogReg) w/ Preprocessed RF-Selected Features	0.4092	0.4364	0.4645	0.4983
05/11/25	Team	Voting Ensemble (tuned SVC, RF, LogReg) w/ Preprocessed RF-Selected Features	0.3692	0.5084	0.5041	0.1051
05/13/25	Team	Stacked Ensemble (Meta: LogReg; Base: tuned SVC, RF, LogReg) w/ Preprocessed RF-Selected Features	0.4098	0.4371	0.4646	0.4949
05/13/25	Team	Voting Ensemble (tuned SVC, RF, LogReg) w/ Preprocessed RF-Selected Features	0.3692	0.5084	0.5041	0.1051

Table 2-2: Employee Fulfillment models' performance on the test set.

Date of Result / Submission to Georgina	Data Source	Predictor Target	Model Version	Balanced Accuracy	Weighted Recall	Weighted F-1	Recall [0]	Highlight if Best
From Page 2 (Our Test Set)								
	Our 20% Test Set	Satisfaction	Satisfaction Stacked Ensemble Page 2 - 05/13/25	0.4388	0.4305	0.4589	0.5255	Our Best (Satisfaction)
	Our 20% Test Set	Fulfillment	Fulfillment Stacked Ensemble Page 2 - 05/13/25	0.4098	0.4371	0.4646	0.4949	Our Best (Fulfillment)
-	-	-	-	-	-	-	-	-
Results from Held-out Data								
	Held-out Data	Satisfaction	Satisfaction Stacked Ensemble Page 2 - 05/13/25	0.3822	0.5699	0.5641	0.1184	
	Held-out Data	Fulfillment	Fulfillment Stacked Ensemble Page 2 - 05/13/25	0.4331	0.4413	0.4673	0.5276	

Table 3: Summary of models' performance for Satisfaction and Fulfillment on test and held-out data.

Note:

Saving our preprocessing pipeline, which included custom transformers, using dill caused crashes. We spent about two days trying to resolve this issue, which delayed our second attempt to improve the model. Because of this, we couldn't submit another model to get a second set of results on the held-out data.

Task 11

(a)

Datasets: Two copies of the 'Company' dataset were used: satisfy_generalizers and fulfill_generalizers.

Target Features:
Satisfaction: Predicting the feature q61 ("Do you like what you do at work?").
Fulfillment: Predicting the feature q62 ("Does your work allow you to bring out your best qualities and abilities?").

The features included responses to survey questions along with metadata from the questionnaire, such as timestamp data (form_start, form_end). Most of the survey questions had only two options, such as question 33, which asks

Do you think that following the daily routine…
1. helps you work calmly
2. Limits your freedom

Some features were ordinal categorical, like question 68, which asked about the respondent's level of position within the company. Lastly, there were multi-label categorical features, where multiple answers could be selected, such as question 74, which asked respondents to select the financial obligations that applied to them (e.g., rent, loans, dependents).

(b)

We are treating this as a classification problem because the target variables (satisfaction and fulfillment) have discrete categories rather than continuous values. The classes for both satisfaction and fulfillment predictions after encoding are:
3 (Yes) → Very satisfied/fulfilled
2 (Rather yes) → Somewhat satisfied/fulfilled
1 (Rather no) → Somewhat dissatisfied/unfulfilled
0 (No) → Very dissatisfied/unfulfilled

This is a multinomial classification because there are more than 2 classes, each employee can only belong to one class, and the model needs to predict which category an instance belongs to.

(c)

We preprocessed the pipeline by separating each feature into four categories: timestamp, categorical binary, ordinal, and multi-label features. For each column transformation, we used a SimpleImputer with the most frequent value to handle any potential missing data. Additionally, we standardized all features with values other than 0 or 1 to ensure compatibility with certain algorithms. We plan to use DBSCAN and SVMs, both of which require standardized data; otherwise, features with larger value ranges might influence the model performance.

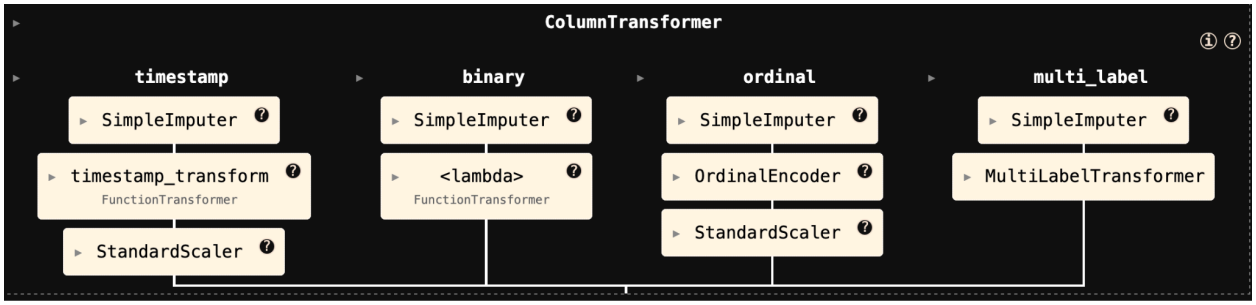

Figure 11-1: Preprocessing Pipeline Diagram

For the form_start and form_end features, we transformed them into two new features: the duration in minutes between the two timestamps, and the time of day the form was submitted (e.g., morning, afternoon, or evening). This transformation may help capture whether the employee completed the form in a rushed or thoughtful manner, as well as whether the time of day reveals any patterns.

For binary categorical features, questions with only two possible responses (e.g., Yes/No), we applied a custom lambda function to convert values from 1/2 to 0/1. This standardization ensures better compatibility with most machine learning algorithms.

For categorical features with a clear order (e.g., 1. Supervisor, 2. Direct or Merchandiser, 3. Ordinary Employee), we used an OrdinalEncoder and then scaled the resulting values to maintain consistency across all features.

Multi-label features, which allowed respondents to select multiple options, resulted in a high number of permutations when one-hot encoded. To manage this, we applied a MultiLabelBinarizer, which reduced the potential 300+ features to just over 100, while preserving all relevant information. Although this still left us with a high-dimensional dataset, we chose to keep the full feature set at this stage to allow for feature selection in the next tasks.

(d)

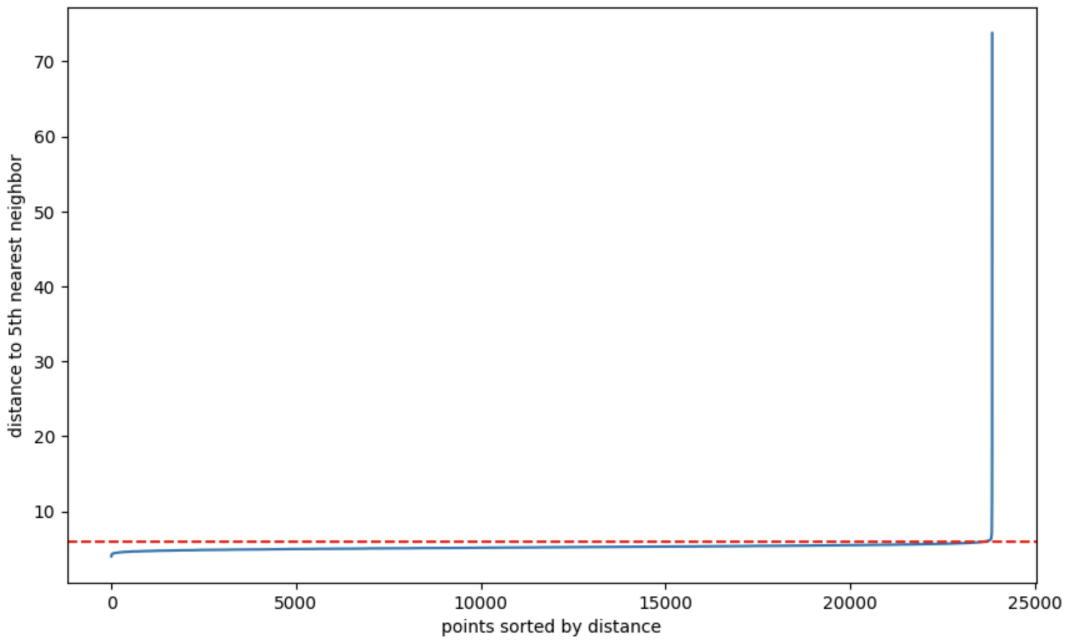

Figure 11-2: DBSCAN Epsilon Selection Plot (k-distance graph).

We dropped a total of 190 data points using DBSCAN, 88 from the Satisfaction dataset and 102 from Fulfillment, leaving 23,795 and 23,737 points, respectively. We chose these based on ***Figure 11-2.***

The graph shows how far each point is from its 5th nearest neighbor, with points sorted by distance. We looked for the “elbow” point in the graph, where the distances start increasing quickly, to help pick the epsilon (eps) value for DBSCAN. Based on this, we chose eps = 5.9 and min_samples = 5 for both datasets.

Although the removed outliers represented only about 0.4% of the data, we believe this small reduction can meaningfully improve model performance. Outliers with unusually large distances to their neighbors can distort classification boundaries and increase the risk of overfitting. Removing them helps preserve data quality while still keeping the overall pattern.

We avoided overly aggressive filtering, since the dataset is relatively small and excessive pruning could eliminate valid instances. For example, lower eps values (3.0 and 4.0) removed hundreds of points, including many likely valid ones. Our chosen parameters aimed to strike a balance: removing clear outliers without discarding useful instances.

(e)

For both the satisfy_generalizers (Satisfaction) and fulfill_generalizers (Fulfillment) tasks, feature selection was applied after the initial preprocessing, starting with 114 features.

Out of the initial 114 preprocessed features, 79 were dropped for each task (114 - 35 = 79). The specific features dropped are those not included in the final selected lists below for each respective task. No new features were added during this dimensionality reduction stage.

For both the satisfy_generalizers and fulfill_generalizers tasks, after initial preprocessing resulted in 114 features, we applied Random Forest Feature Selection to reduce dimensionality and identify the most impactful features. This method was chosen over alternatives like PCA and Hierarchical Clustering (detailed in Task 12).

After applying Random Forest feature selection independently, 35 features remained for the satisfy_generalizers task, and 35 features remained for the fulfill_generalizers task.

Selected Features: **satisfy_generalizers** - 35 features:
['timestamp__duration_minutes', 'timestamp__part_of_day', 'binary__q01', 'binary__q08', 'binary__q09', 'binary__q22', 'binary__q26', 'binary__q28', 'binary__q33', 'binary__q41', 'binary__q42', 'binary__q46', 'binary__q49', 'binary__q60', 'ordinal__q68', 'ordinal__q69', 'ordinal__q70', 'multi_label__q63_1', 'multi_label__q63_4', 'multi_label__q64_1', 'multi_label__q64_3', 'multi_label__q64_6', 'multi_label__q64_7', 'multi_label__q64_8', 'multi_label__q65_2', 'multi_label__q65_3', 'multi_label__q65_7', 'multi_label__q66_1', 'multi_label__q66_2', 'multi_label__q66_3', 'multi_label__q66_4', 'multi_label__q66_7', 'multi_label__q67_1', 'multi_label__q67_2', 'multi_label__q67_5']

	feature	importance
0	timestamp__duration_minutes	0.047813
90	multi_label__q66_3	0.030147
66	ordinal__q70	0.025618
74	multi_label__q64_1	0.023737
80	multi_label__q64_8	0.021228
34	binary__q33	0.020996
65	ordinal__q69	0.019368
1	timestamp__part_of_day	0.017706
64	ordinal__q68	0.016848
67	multi_label__q63_1	0.016686

Figure 11-3: Top 10 Feature Importances (Satisfaction)

Selected Features: **fulfill_generalizers** - 35 features
['timestamp__duration_minutes', 'timestamp__part_of_day', 'binary__q06', 'binary__q08', 'binary__q09', 'binary__q22', 'binary__q26', 'binary__q33', 'binary__q34', 'binary__q41', 'binary__q42', 'binary__q46', 'binary__q49', 'binary__q60', 'ordinal__q68', 'ordinal__q69', 'ordinal__q70', 'multi_label__q63_1', 'multi_label__q63_4', 'multi_label__q64_1', 'multi_label__q64_3', 'multi_label__q64_6', 'multi_label__q64_7', 'multi_label__q64_8', 'multi_label__q65_2', 'multi_label__q65_7', 'multi_label__q66_1', 'multi_label__q66_2', 'multi_label__q66_3', 'multi_label__q66_4', 'multi_label__q66_7', 'multi_label__q67_1', 'multi_label__q67_2', 'multi_label__q67_5', 'multi_label__q74_4']

	feature	importance
0	timestamp__duration_minutes	0.048950
90	multi_label__q66_3	0.029916
66	ordinal__q70	0.025995
74	multi_label__q64_1	0.022226
65	ordinal__q69	0.019945
80	multi_label__q64_8	0.019315
34	binary__q33	0.019094
64	ordinal__q68	0.017693
67	multi_label__q63_1	0.017256
1	timestamp__part_of_day	0.017126

Figure 11-4: Top 10 Feature Importances (Fulfillment)

(f)

Evaluation Metrics:

Our primary tuning metric was **balanced accuracy**, chosen due to the significant class imbalance in both datasets, there were many more satisfied/fulfilled employees than dissatisfied/unfulfilled ones. Although standard accuracy can be misleading with imbalanced data, we still reported it for reference, but did not rely on it for tuning.

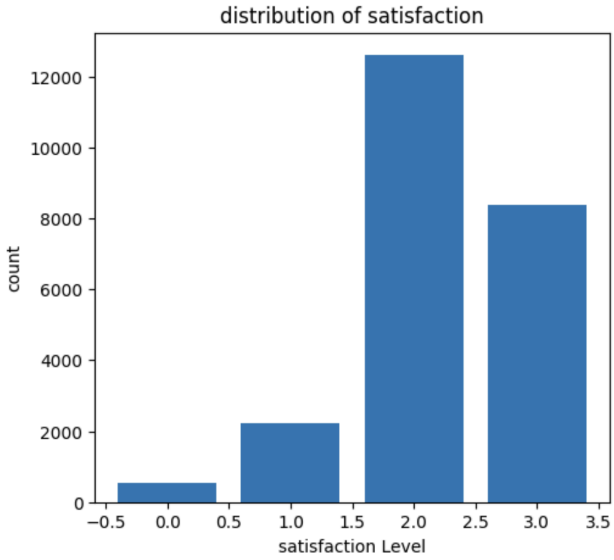

Figure 11-5: Distribution of Satisfaction Classes

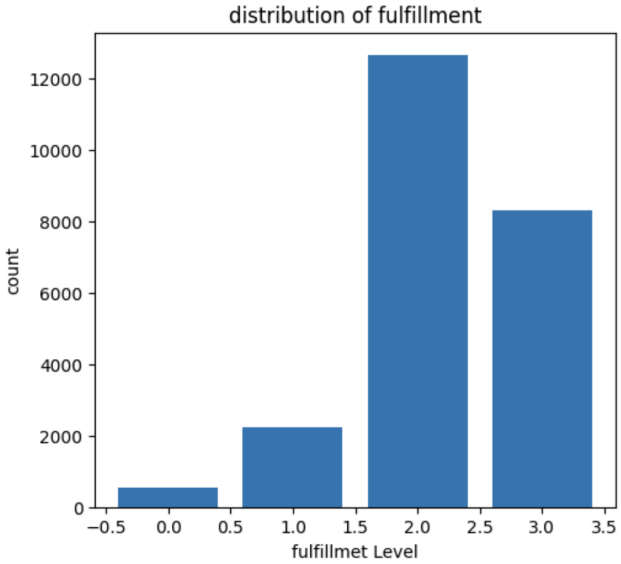

Figure 11-6: Distribution of Fulfillment Classes

We also computed weighted recall and weighted F1-score to better capture model performance across all classes. In our final evaluation on the separated test set, we computed specific recall for classes 0 and 1, which represent unhappy employees.

Identifying discontentment is important in a workplace setting, as unhappy employees can reduce productivity, delay work, and negatively impact the team. Because of this, we prioritized recall to minimize false negatives, cases where the model fails to detect discontentment.

Algorithms:
We evaluated a few base learners:
Logistic Regression (LR)
Support Vector Classifier (SVC),
Random Forest (RF),
Gradient Boosting (GB),
Multi-Layer Perceptron (MLP)

In addition, we explored two ensemble strategies:

1) Stacking Classifier using LR, RF, and SVC as base learners with LR as the meta-learner.
2) Voting Classifier with soft voting across LR, RF, and SVC.

Our ensemble was designed to combine strengths from diverse model types, linear, non-linear, tree-based, and neural network-based. We selected the top three performers among the base models and compared the performance of the voting classifier against the stacking classifier to get the best-performing and robust model for each task.

(g)

Instead of creating a separate validation set, parameter tuning for both tasks was performed using 3-fold cross-validation, directly on the preprocessed training data (with no outliers and reduced dimensionality). We used RandomizedSearchCV (for faster computation) to find optimal hyperparameters for base algorithms. The primary metric for the search was balanced_accuracy. This tuning process was executed independently for each target variable, and the cross-validation results, in specific the balanced accuracy scores, gave us our final choices for model hyperparameters and the selection of base models for the ensemble methods.

To facilitate this process, we created a tune_model function. This function took the base learner model object, its parameter grid, the number of cross-validation folds, and the scoring metric as input. The function then performed the tuning using RandomizedSearchCV, returned the best model found, and printed the cross-validation performance metrics for that best model.

```python
def tune_model(base_model, param_grid, X, y, name, cv=3, scoring_metric='balanced_accuracy'):
    random_search = RandomizedSearchCV(
        base_model,
        param_distributions=param_grid,
        n_iter=20,
        cv=cv,
        scoring=scoring_metric,
        random_state=42,
        n_jobs=-1
    )

    random_search.fit(X, y)
    best_model = random_search.best_estimator_
    best_primary_score = random_search.best_score_

    accuracy = cross_val_score(best_model, X, y, cv=cv, scoring='accuracy').mean()
    balanced_acc = cross_val_score(best_model, X, y, cv=cv, scoring='balanced_accuracy').mean()
    recall = cross_val_score(best_model, X, y, cv=cv, scoring='recall_weighted').mean()
    f1 = cross_val_score(best_model, X, y, cv=cv, scoring='f1_weighted').mean()

    print(f"Best {scoring_metric}: {best_primary_score:.4f}")
    print(f"Regular accuracy: {accuracy:.4f}")
    print(f"Balanced accuracy: {balanced_acc:.4f}")
    print(f"Weighted recall: {recall:.4f}")
    print(f"Weighted F1: {f1:.4f}")

    return best_model
```

Figure 11-7: tune_model Function Code Snippet

Satisfaction: Base models fine-tuned using the tune_model function:

```python
LR = LogisticRegression(random_state=42)
lr_rf_best_model = tune_model(LR, param_grid_LR, X_rf_filtered, y_satisfy_encoded,
                              "Logit Regression using RF features")
```

```
Best balanced_accuracy: 0.4646
Regular accuracy: 0.4447
Balanced accuracy: 0.4646
Weighted recall: 0.4447
Weighted F1: 0.4636
```

```python
RF = RandomForestClassifier(random_state=42)
rf_rf_best_model = tune_model(RF, param_grid_RF, X_rf_filtered, y_satisfy_encoded,
                              "Random Forest using RF features")
```

```
Best balanced_accuracy: 0.4075
Regular accuracy: 0.5628
Balanced accuracy: 0.4075
Weighted recall: 0.5628
Weighted F1: 0.5625
```

```python
svc_model = SVC(random_state=42)
svc_rf_best_model = tune_model(svc_model, param_grid_SVC, X_rf_filtered, y_satisfy_encoded,
                              "SVC using RF features")
```

```
Best balanced_accuracy: 0.4647
Regular accuracy: 0.4599
Balanced accuracy: 0.4647
Weighted recall: 0.4599
Weighted F1: 0.4802
```

Figure 11-8: Base Model Tuning Results (Satisfaction)

Fulfilment: base models fine-tuned using the tune_model function:

```
LR = LogisticRegression(random_state=42)
lr_rf_best_model = tune_model(LR, param_grid_LR, X_rf_filtered, y_fulfill_encoded,
                              "Logit Regression using RF features")
```

```
Best balanced_accuracy: 0.4514
Regular accuracy: 0.4423
Weighted recall: 0.4423
Weighted F1: 0.4644
```

```
RF = RandomForestClassifier(random_state=42)
rf_rf_best_model = tune_model(RF, param_grid_RF, X_rf_filtered, y_fulfill_encoded,
                              "Random Forest using RF features")
```

```
Best balanced_accuracy: 0.4035
Regular accuracy: 0.5628
Weighted recall: 0.5628
Weighted F1: 0.5623
```

```
svc_model = SVC(random_state=42)
svc_rf_best_model = tune_model(svc_model, param_grid_SVC, X_rf_filtered, y_fulfill_encoded,
                              "SVC using RF features")
```

```
Best balanced_accuracy: 0.4521
Regular accuracy: 0.4359
Weighted recall: 0.4359
Weighted F1: 0.4595
```

Figure 11-9: Base Model Tuning Results (Fulfillment)

Based on these cross-validation tuning results, Logistic Regression, Random Forest, and SVC were selected as the base learners for our ensemble models. They achieved the highest balanced accuracy scores among the base learner models.

Task 12

In deciding the approach for the feature sets to train our algorithms on for both satisfy_generalizers and fulfill_generalizers, we considered several strategies. The main approaches evaluated were: using the Original full set of 114 preprocessed features, Random Forest Feature Selection, PCA for dimensionality reduction, and Hierarchical Clustering.

We compared these approaches by having the data preprocessed up to the feature selection stage, and then trained and tested each resulting feature set using a consistent RandomForest algorithm. We ran 3-cross-validation to get the best mean metric for their respective approaches. We compared their metrics across balanced accuracy, recall_weighted, and f1_weighted to see which feature selection strategy performed the best.

For satisfaction target:

	accuracy	balanced_accuracy	recall_weighted	f1_weighted
Original	0.591807	0.385405	0.591807	0.581389
PCA	0.598501	0.329246	0.598501	0.558425
RandomForest	0.560692	0.410087	0.560692	0.560782
Hierarchical	0.550882	0.359309	0.550882	0.543010

Figure 12-1: Feature Selection Method Comparison (Satisfaction)

Random Forest Feature Selection showed the strongest performance on our primary metric with a balanced accuracy of 0.4101 and an f1_weighted score of 0.5608.
PCA achieved a balanced accuracy of 0.3292 and an f1_weighted of 0.5584. While PCA sometimes showed slightly higher raw accuracy (0.5985) than RF-selected features (0.5607), its balanced accuracy was lower.
Hierarchical Clustering provided a balanced accuracy of 0.3593 and an f1_weighted of 0.5430.

For fulfillment target:

	accuracy	balanced_accuracy	recall_weighted	f1_weighted
Original	0.590386	0.382051	0.590386	0.579890
PCA	0.593757	0.327747	0.593757	0.555395
RandomForest	0.560222	0.406055	0.560222	0.560410
Hierarchical	0.535030	0.344756	0.535030	0.525759

Figure 12-2: Feature Selection Method Comparison (Fulfillment)

Random Forest Feature Selection again led with a balanced accuracy of 0.4061 and an f1_weighted score of 0.5604.
PCA resulted in a balanced accuracy of 0.3277 and an f1_weighted of 0.5554.
Hierarchical Clustering provided a balanced accuracy of 0.3448 and an f1_weighted of 0.5258.

Although using the original full set of features or PCA sometimes gave slightly better results for accuracy or weighted recall, Random Forest Feature Selection gave the highest balanced accuracy for both satisfaction and fulfillment predictions. Since balanced accuracy is the most important metric for us, we chose Random Forest

Feature Selection as our final method. We used the mean feature importance as a threshold and removed features that scored below it, as they were likely less useful. This method reduced the number of features the most, by 69%, compared to all other approaches.

An advantage of RF feature selection is interpretability, giving insights into which features contribute most to predicting employee satisfaction and fulfillment, which can be important in a company setting.

After completing feature selection, we evaluated several models and proceeded to our two ensemble strategies: Stacking and Voting, using the top three base models mentioned at the end of Task 11. We chose these two ensembles because we thought combining weak learners would allow for a stronger model in predicting our target class. In the end, for both satisfaction and fulfillment predictions, the Stacking Ensemble using Logistic Regression as the meta-learner was selected. This choice was supported by 95% confidence intervals (CIs) for balanced accuracy from 5-fold cross-validation on our test set.

Satisfaction Prediction:
Stacking Ensemble: Mean balanced accuracy of 0.4348 (95% CI: 0.4134, 0.4562). On our 05/13/25 test, it achieved a balanced accuracy of 0.4388 and a Class 0 recall of 0.5255.

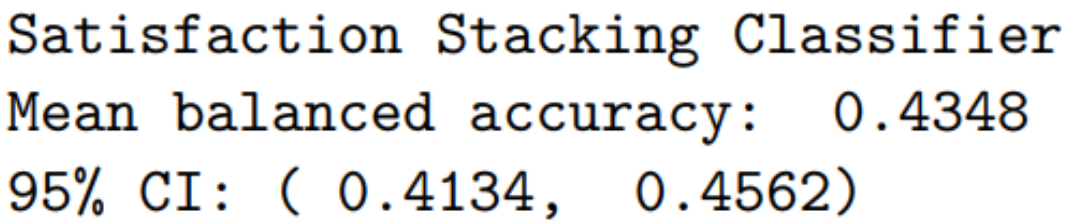

Figure 12-3: Satisfaction Stacking Classifier Performance 95% CI

Voting Ensemble: Mean balanced accuracy of 0.3909 (95% CI: 0.3705, 0.4113). Its Class 0 recall was lower at 0.2044.

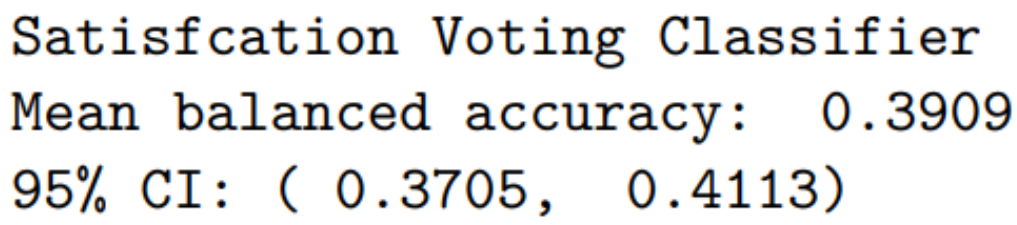

Figure 12-4: Satisfaction Voting Classifier Performance 95% CI

The Stacking Ensemble was chosen because its 95% CI for balanced accuracy was higher and did not overlap with the Voting Ensemble's, indicating statistically significantly better performance. Additionally, its superior Class 0 recall aligned with our goal.

Fulfillment Prediction:
Stacking Ensemble: Mean balanced accuracy of 0.4081 (95% CI: 0.3758, 0.4405).
On our 05/13/25 test, it achieved a balanced accuracy of 0.4098 and a Class 0 recall of 0.4949.

```
Fulfillment Stacking Classifier
Mean balanced accuracy:  0.4081
95% CI: ( 0.3758,  0.4405)
```

Figure 12-5: Fulfillment Stacking Classifier Performance 95% CI

Voting Ensemble: Mean balanced accuracy of 0.3883 (95% CI: 0.3651, 0.4114).
Its Class 0 recall was considerably lower at 0.1051.

```
Fulfillment Voting Classifier
Mean balanced accuracy:  0.3883
95% CI: ( 0.3651,  0.4114)
```

Figure 12-6: Fulfillment Voting Classifier Performance 95% CI

The Stacking Ensemble was chosen for the fulfillment task because it had higher balanced accuracy and much better recall for Class 0.

In summary, the Stacking Ensemble was selected for both tasks due to its higher or comparable balanced accuracy (supported by 95% CIs) and its significantly better ability to identify the most discontent employees (Class 0 recall), which was a crucial factor.

Task 13

(a)

Taking a look at the dataset, there seems to be no correlation between our features and the target class. After running a feature importance analysis using Random

Forest feature selection, our highest mean importance was at 0.04. This means that a lot of the features were uncorrelated with our target class. Taking a deeper look at each question, we thought a lot of these questions didn't help predict our target class for both satisfaction and fulfillment:

```
Question 5: You can most likely be called…
Answers
 1. practical person
 2. fictional, romantic
```

Figure 13-1: Example Uninformative Survey Question 5

```
Question 10: When you read for your pleasure, you like…
Answers
 1. unusual, original manner of presentation
 2. when writers express their thoughts clearly
```

Figure 13-2: Example Uninformative Survey Question 10

These questions that we thought didn't relate to employee satisfaction and fulfillment made it difficult to learn from instances. Looking at the heavily imbalanced dataset, predicting whether an employee was ever dissatisfied or unfulfilled at their job was our priority.

Evaluating our three approaches: random forest feature selection, PCA, and hierarchical clustering, we thought we would be able to train strong models as they'd shrink the size of our features and group similar features together, creating the most important features in determining satisfaction and fulfillment.

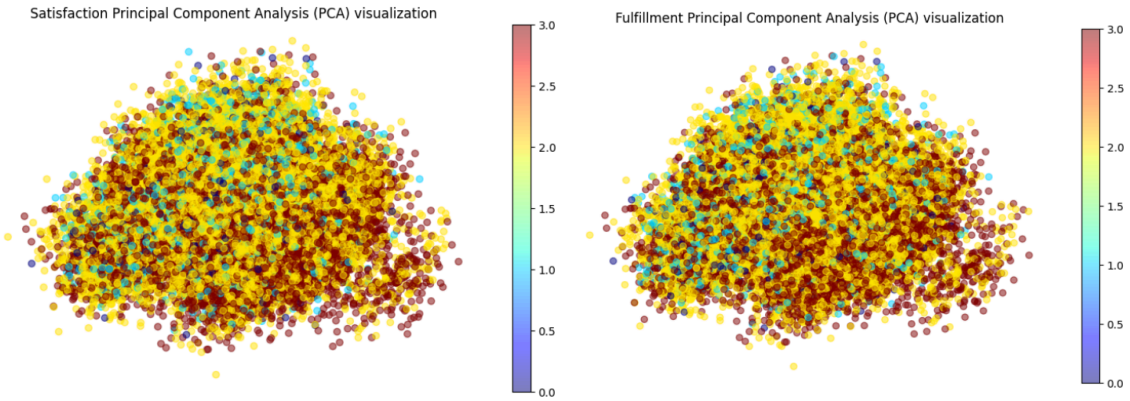

Figure 13-3: PCA 2D visualization (Satisfaction and Fulfillment)

PCA visualizes the first two components for satisfaction(left) and fulfillment(right): The colors represent the different classes, and we can see that there's no clear separation or clustering among the classes.

	feature	importance
0	timestamp__duration_minutes	0.047813
90	multi_label__q66_3	0.030147
66	ordinal__q70	0.025618
74	multi_label__q64_1	0.023737
80	multi_label__q64_8	0.021228
34	binary__q33	0.020996
65	ordinal__q69	0.019368
1	timestamp__part_of_day	0.017706
64	ordinal__q68	0.016848
67	multi_label__q63_1	0.016686

	feature	importance
0	timestamp__duration_minutes	0.048950
90	multi_label__q66_3	0.029916
66	ordinal__q70	0.025995
74	multi_label__q64_1	0.022226
65	ordinal__q69	0.019945
80	multi_label__q64_8	0.019315
34	binary__q33	0.019094
64	ordinal__q68	0.017693
67	multi_label__q63_1	0.017256
1	timestamp__part_of_day	0.017126

Figure 13-4: Top 10 Feature Importances (Satisfaction and Fulfillment)

Random Forest feature selection lists out the features with the most importance in predicting our target class: satisfaction(left) and fulfillment(right).

These approaches didn't yield significant results that would help predict our target class. PCA created a giant cluster with no separation among classes, while our top feature importance for our Random Forest approach had the best feature contributing to only ~4% in predicting our target class. We believe because of the class imbalance of discontent and dissatisfied employees, making up only about ~15% of the dataset, it was difficult to predict the target classes.

As for our learners, we used a variety of models, but despite the diversity, most of them struggled to perform well due to the nature of the dataset. Even with hyperparameter tuning and ensemble methods, improvements were small. Simple logistic regression models perform just as well as more complex ones, like MLPs or random forests. This suggests that even powerful learners can't compensate for low quality data and uninformative features.

(b)

No, we think we still have a lot of room to improve on. Having a balanced accuracy of below 50% means that we cannot accurately predict the satisfaction or fulfillment of an employee based on such low metrics. While recall for the dissatisfied and unfulfilled, class 0, showed improvement, it wasn't close to 90%, meaning most of the discontented employees might still be missed by the predictor. Though this score was above the random baseline of 0.25 for our four-class classification, it is not high enough to be considered reliable. While recall for class 0 showed some improvement, it

was still far from ideal, meaning most unhappy employees may still be missed. We trained several base models that reached around 50% accuracy on different evaluation metrics and tested two ensemble methods, voting and stacking. Although stacking performed slightly better than voting, the overall performance suggests that more improvements are needed to be considered a trustworthy model.

(c)

We were disappointed by the lack of correlation between the feature and the limited examples of class 0 (very discontent employees), which negatively impacted our results. This lack of correlation made the hierarchical clustering feature selection less effective, as it relied on a distance matrix based on correlation, leading to poor performance. Additionally, the overall balance accuracy was lower than expected. The class imbalance further constrained our models from performing well across all classes.

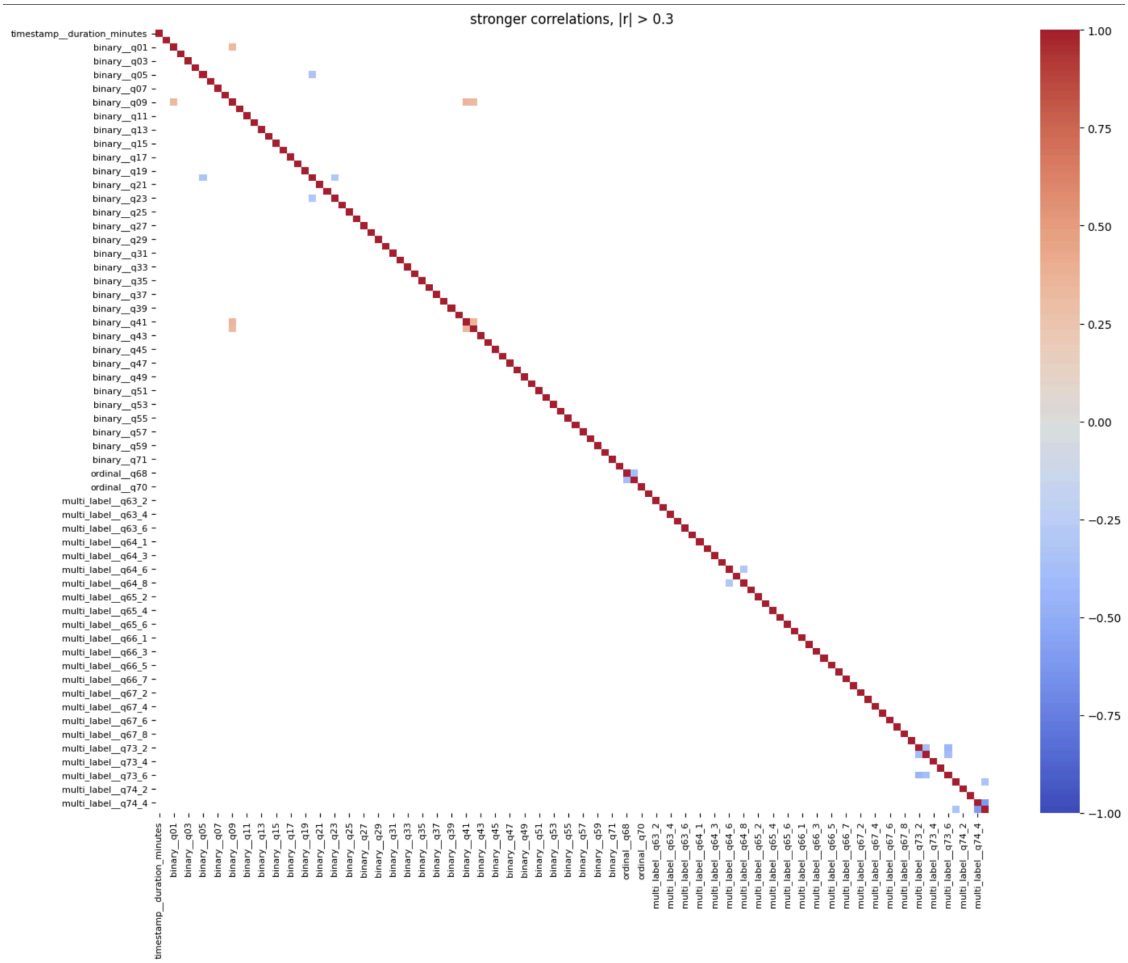

Figure 13-5: Feature correlation heatmap showing the few strong relationships ($|r|$>0.3) in the preprocessed dataset.

A surprising finding was that PCA required a high number of principal components, ~90 components, to capture 95% of the cumulative variance. We had expected that PCA would reduce the dimensionality significantly while preserving most of the variance, but this wasn't the case.

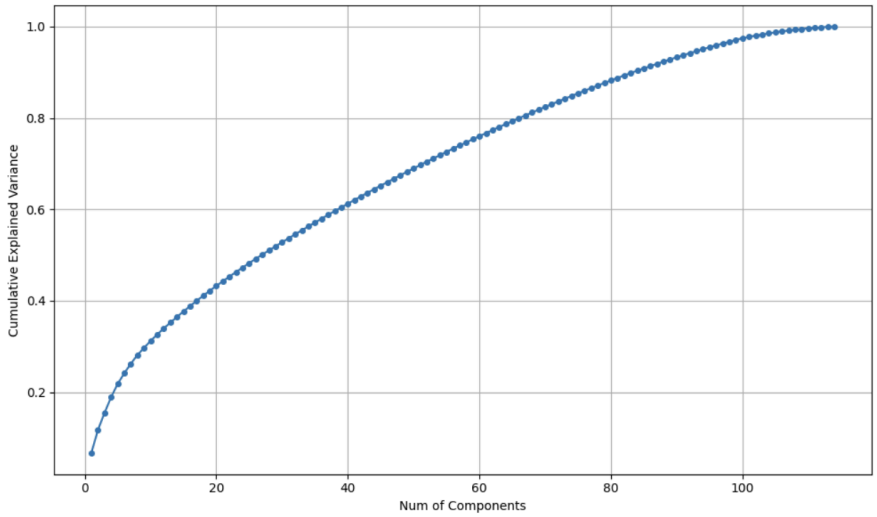

Figure 13-6: PCA cumulative Explained Variance Plot for Satisfaction (Fulfillment had a similar graph.)

We also believed that Random Forest feature selection would not only reduce dimensions but also improve the metrics compared to PCA or the preprocessed dataset with the full feature set, but this didn't happen. RF feature selection only improved balanced accuracy, while all other metrics remained similar to those of PCA. All approaches didn't yield the best results, as clustering didn't help separate the classes well, and the random forest feature selection had the highest feature importance of only 0.04 in predicting the target class. Another surprise was the fact that simple logistic regression models perform just as well as more complex ones, like MLPs or random forests.

However, one thing that pleased us was that the model did not overfit the training set, as the evaluation results on the training set were similar to those on the test set. Another positive aspect was that the stacking classifier outperformed the voting classifier, as we had expected. Lastly, we were pleased to see our decomposed features: 'timestamp__duration_minutes', 'timestamp__part_of_day', ranked among the top features in terms of importance for both tasks.

Overall, our model performed similarly on both the 20% test set and the held-out set, showing consistent results. For both satisfaction and fulfillment predictions, the Stacked Ensemble achieved our highest balanced accuracy and recall for class 0. While performance dropped slightly on the held-out data for satisfaction, fulfillment remained stable. This suggests that our model generalizes better for fulfillment than satisfaction, although both tasks still need improvement.