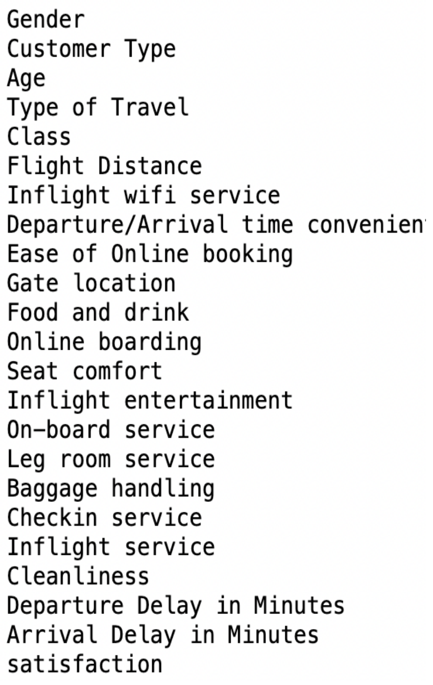
ML PROJECT REPORT

PROBLEM OF INTEREST - Predict passenger satisfaction for an Airline and I'm planning to use the dataset Airline Passenger Satisfaction from Kaggle using Binary Classification. This Dataset has two files which are already divided into train and test dataset with test data being 20% of the total data.

EXPLORATORY DATA ANALYSIS -

- The Dataset contains 103904 rows and 24 features(+1 column for label).
- Among all features, 5 are object type features. Others are numerical type (int,float) features.
- All features have 103904 values except Arrival Delay in Minutes(103594 values). It has 310 null values in train data
- There are no duplicate rows in training data and test data
- All features have 25976 non-null values except Arrival Delay in Minutes(25893 values). It has 83 null values. In Testing data.
- Features are given below. Satisfaction is the label.



```
Gender
Customer Type
Age
Type of Travel
Class
Flight Distance
Inflight wifi service
Departure/Arrival time convenien
Ease of Online booking
Gate location
Food and drink
Online boarding
Seat comfort
Inflight entertainment
On-board service
Leg room service
Baggage handling
Checkin service
Inflight service
Cleanliness
Departure Delay in Minutes
Arrival Delay in Minutes
satisfaction
```

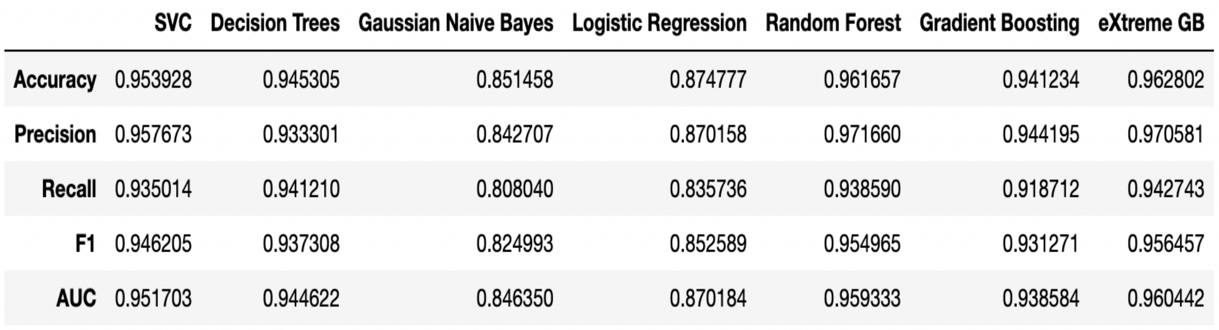
- The label 'Satisfaction' has two values "Satisfied" and "Neutral/Dissatisfied". "Neutral/Dissatisfied" can be considered as "Not Satisfied" and this will be a Binary Classification problem.

PREPROCESSING DATA -

- We use LabelEncoder to encode Satisfaction label values to 0 and 1. 0 is mapped to ‘Neutral or dissatisfied’ and 1 is mapped to ‘Satisfied’.
- We use OneHotEncoder() to encode categorical feature - 'Gender', 'Customer Type', 'Type of Travel', 'Class'.
- For features with only 2 labels, drop one column as the other one has both values 0 and 1.
- Since different features have different ranges of values, we use StandardScaler to standardize them.
- Replaced Null values with the mean for Arrival Delay in Minutes feature in both train and test data.

COMPARISON OF DIFFERENT MODELS -

- Compare the Accuracy, Precision, Recall, F1 score, AUC for different models which are modeled with their default parameters.
- The models used here are -
    - SVC - Default parameters - kernel='rbf', degree=3
    - Decision Trees - Default parameters - criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1
    - Gaussian Naive Bayes - Default parameters - priors=None, var_smoothing=1e-09
    - Logistic Regression - Default parameters - penalty - 'l2': add a L2 penalty term
    - Random Forest - Default parameters - n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1
    - Gradient Boosting - Default parameters - loss='log_loss', learning_rate=0.1, n_estimators=100, min_samples_split=2, min_samples_leaf=1, max_depth=3
    - eXtreme Gradient Boosting - Default parameters - max_depth=3, learning_rate=0.1, n_estimators=100



	SVC	Decision Trees	Gaussian Naive Bayes	Logistic Regression	Random Forest	Gradient Boosting	eXtreme GB
Accuracy	0.953928	0.945305	0.851458	0.874777	0.961657	0.941234	0.962802
Precision	0.957673	0.933301	0.842707	0.870158	0.971660	0.944195	0.970581
Recall	0.935014	0.941210	0.808040	0.835736	0.938590	0.918712	0.942743
F1	0.946205	0.937308	0.824993	0.852589	0.954965	0.931271	0.956457
AUC	0.951703	0.944622	0.846350	0.870184	0.959333	0.938584	0.960442

-

- Evaluate a score by cross-validation. It uses default 5-fold cross validation((Stratified)KFold). We take the mean of the scores of the 5 validation sets from the cross_val_score function.
- We can see that Random Forest and eXtreme GB classifier performs better than other classifiers on the train data. They have better Accuracy scores.
- The F1 score is a measure of a model's accuracy that takes into account both precision and recall. Random Forest and eXtreme GB classifier have higher F1 scores compared to others.

FEATURE ENGINEERING -

- Used SelectKBest to find the 'k' most important features before training the data using any model. We get the feature names but we don't get the score for them.
- With k=15, we get these features as important - 'Age', 'Flight Distance', 'Inflight wifi service', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board service', 'Leg room service', 'Checkin service', 'Cleanliness', 'Departure Delay in Minutes','Type of Travel', 'Class_Business', 'Class_Eco'
- To check the important features with highest scores, after fitting the train data with the model, we get the scores from inbuilt functions of model.feature_importances_ and model.coef_[0]

HYPERPARAMETER TUNING -

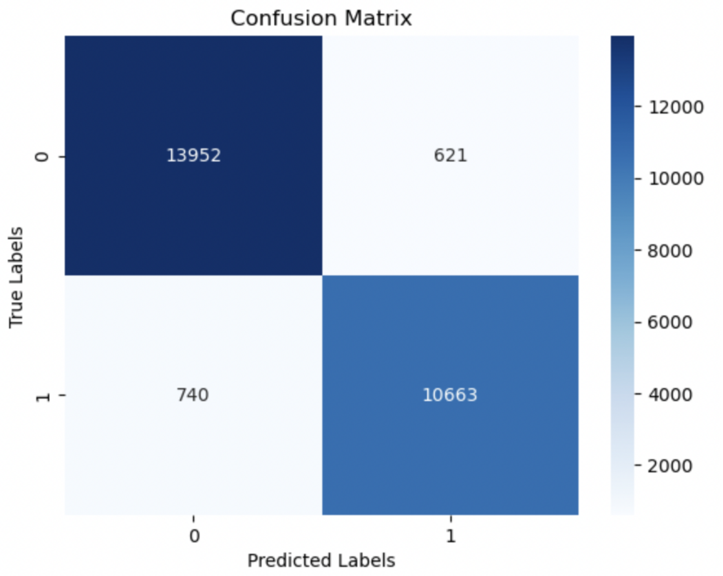
- Used GridSearchCV - specify a list of hyperparameters and a performance metric, the algorithm works through all possible combinations to determine the best fit.

- Decision trees -
    - Parameters tested = {
    - "max_depth": [3, 5, 7, 9, 11, 13], "max_features": [2, 5, 8, 10],
    - "min_samples_split": [2, 5, 7, 9], "criterion": ["gini", "entropy"]}
    - Best parameters: {'criterion': 'entropy', 'max_depth': 13, 'max_features': 10, 'min_samples_split': 2}
    - Mean cross-validated accuracy score of the best_estimator: 0.9478
- Logistic Regression -
- Parameters tested = {
    - "C": [0.001, 0.01, 0.1, 1.],"penalty": ["l1", "l2"]
    - Best parameters {'C': 0.1, 'penalty': 'l1'}
    - Mean cross-validated accuracy score of the best_estimator: 0.8748
- Random Forest -
    - Parameters tested = {
    - 'n_estimators': [100, 200], 'max_depth': [None, 5, 10],
    - 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]}
    - Best parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200}
    - Mean cross-validated accuracy score of the best_estimator: 0.9623
- Gradient Boosting -
    - parameters_gb = {
    - 'max_depth': [3, 5, 7, 9], 'n_estimators': [5, 10, 15, 20, 25, 50, 100],
    - 'learning_rate': [0.01, 0.05, 0.1]
    - Best parameters: {'learning_rate': 0.1, 'max_depth': 9, 'n_estimators': 100}
    - Mean cross-validated accuracy score of the best_estimator: 0.9634
- eXtreme GB -
    - parameters_xgb = {
    - 'max_depth': [3, 5, 7, 9], 'n_estimators': [5, 10, 15, 20, 25, 50, 100],
    - 'learning_rate': [0.01, 0.05, 0.1]
    - Best parameters {'learning_rate': 0.1, 'max_depth': 9, 'n_estimators': 100}
    - Mean cross-validated accuracy score of the best_estimator: 0.9638
- AdaBoost -
    - parameters = {

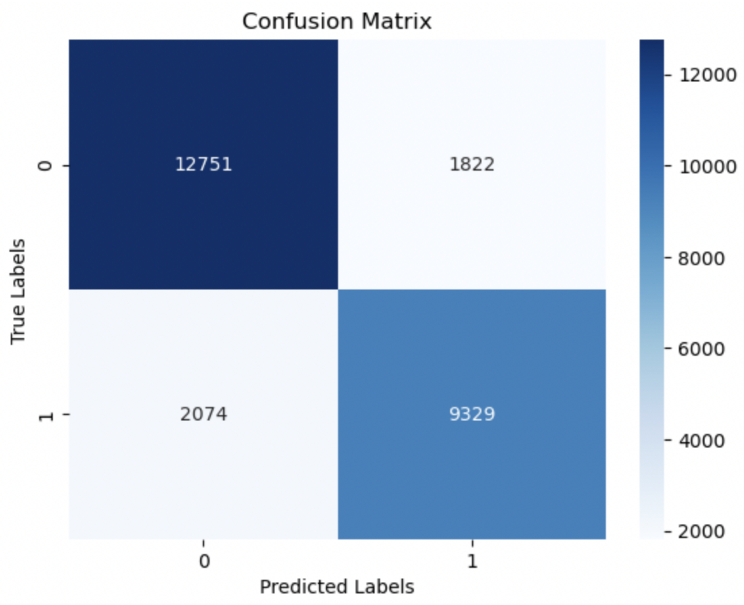
- "n_estimators": [5, 10, 15, 20, 25, 50, 75, 100],
- "learning_rate": [0.001, 0.01, 0.1, 1.],
- Best parameters {'learning_rate': 1.0, 'n_estimators': 75}
- Mean cross-validated accuracy score of the best_estimator: 0.9291

MODELING WITH SELECTED IMPORTANT FEATURES AND HYPERPARAMETER TUNING -

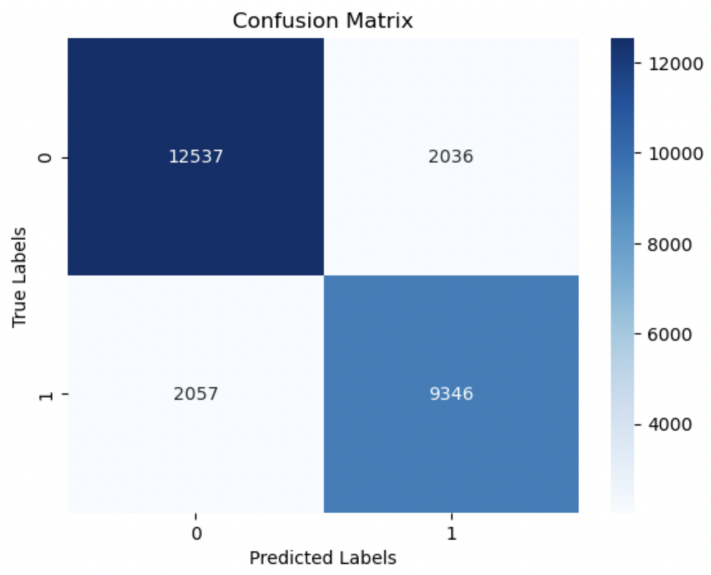
- We model the data with selected most important features which have a score greater than a certain threshold and use the best parameters we got from hyperparameter tuning.
- Confusion Matrices for the models -
  - Decision Tree -



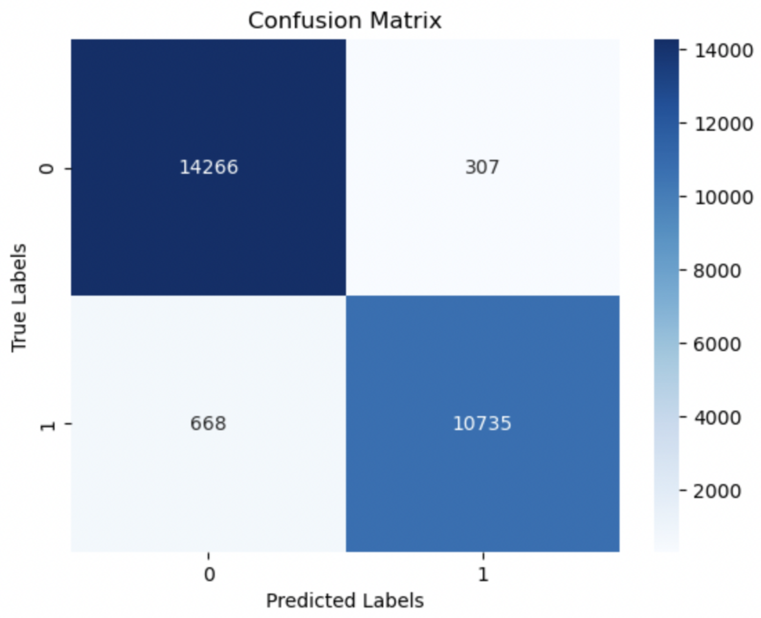
  -
  - Logistic Regression -



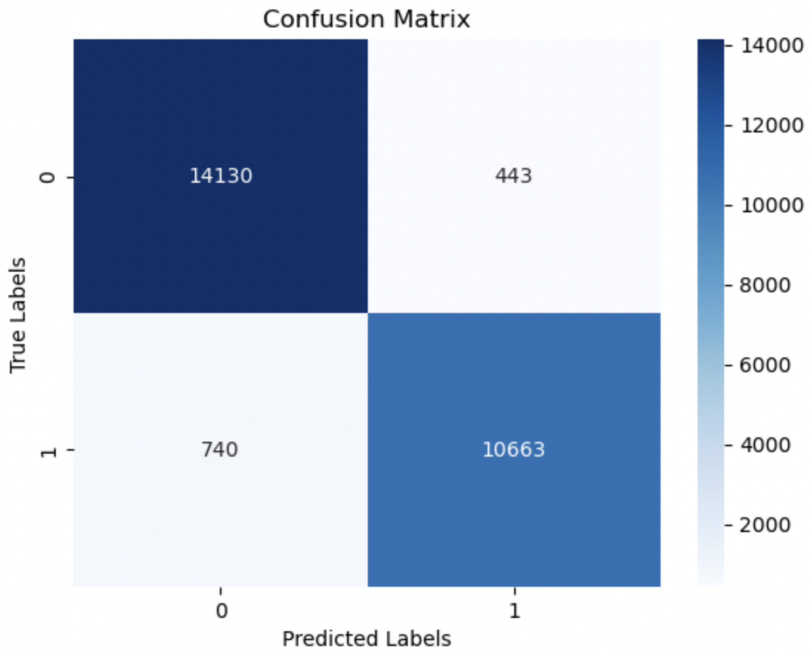
-
- Gaussian Naive Bayes -



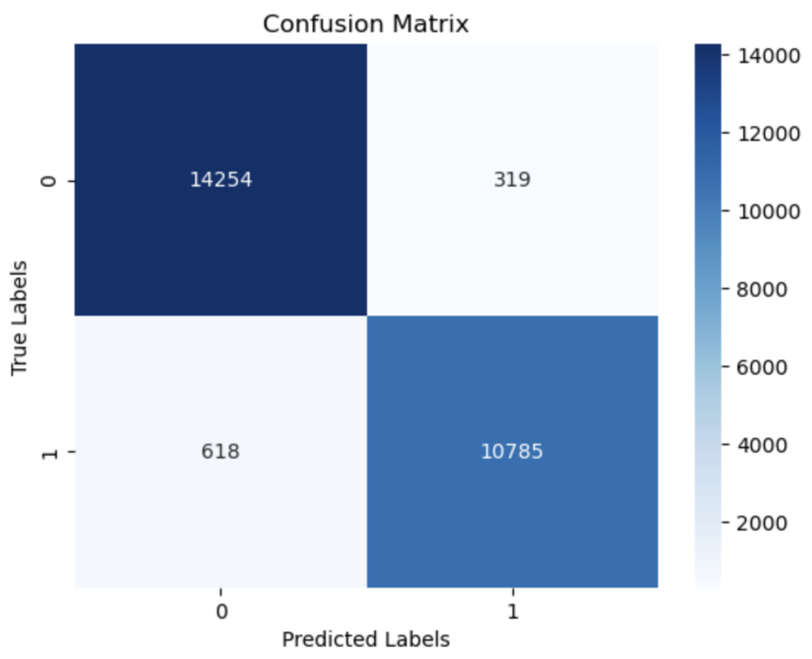
-
- Random Forest -



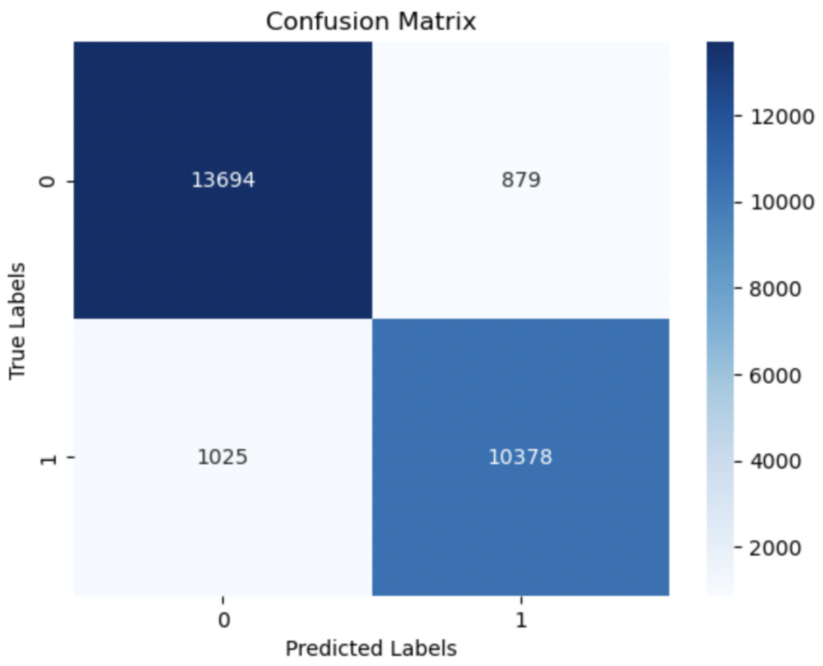
-
- Gradient Boosting -



-
- eXtreme GB -



-
- Ada Boost -



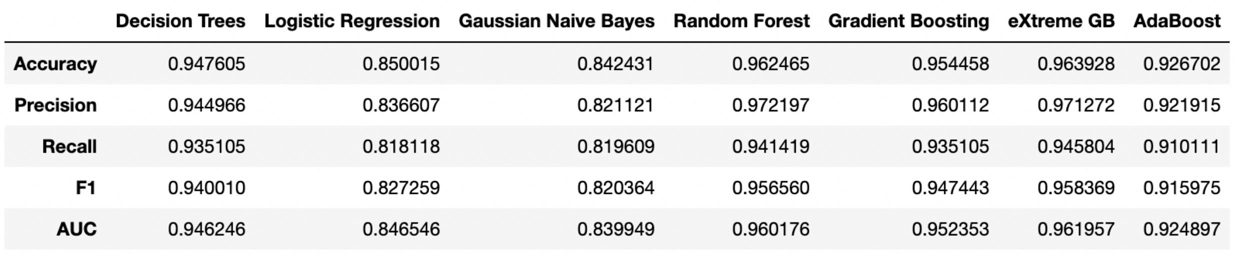
-

EVALUATION -

- Comparison of different models on train, validation dataset -

	Decision Trees	Logistic Regression	Gaussian Naive Bayes	Random Forest	Gradient Boosting	eXtreme GB	AdaBoost
Accuracy	0.946691	0.852439	0.848118	0.962523	0.952157	0.963784	0.927788
Precision	0.943560	0.836857	0.825439	0.974194	0.955335	0.973258	0.924066
Recall	0.932393	0.819100	0.823698	0.938523	0.932415	0.942321	0.907984
F1	0.938053	0.827912	0.824558	0.955933	0.943560	0.957537	0.915949
AUC	0.944894	0.848506	0.845245	0.959643	0.949619	0.961259	0.925459

- Comparison of different model on test dataset -



	Decision Trees	Logistic Regression	Gaussian Naive Bayes	Random Forest	Gradient Boosting	eXtreme GB	AdaBoost
Accuracy	0.947605	0.850015	0.842431	0.962465	0.954458	0.963928	0.926702
Precision	0.944966	0.836607	0.821121	0.972197	0.960112	0.971272	0.921915
Recall	0.935105	0.818118	0.819609	0.941419	0.935105	0.945804	0.910111
F1	0.940010	0.827259	0.820364	0.956560	0.947443	0.958369	0.915975
AUC	0.946246	0.846546	0.839949	0.960176	0.952353	0.961957	0.924897

- We observe that Gradient Boosting and eXtremeGB have the highest accuracy from all other models. Their F1 scores are the highest as well.

RESULTS -

- Before Feature selection and hyperparameter tuning, we observe that XGB has the highest accuracy with 0.9628, followed by Random Forest with 0.9616 accuracy. Their F1 scores are the highest and almost same.
- After feature selection and hyperparameter tuning, we observe that XGB and Random Forest have the highest accuracy of 0.9639 and 0.9624 respectively from all other models. Their F1 scores are the highest as well.
- The scores before and after are almost same but the execution of fitting model is faster in the later case.
- Before Hyperparameter Tuning and feature selection (Training time in secs)
  - DT - 7.236382007598877
  - LR - 2.6797268390655518
  - Gaussian NB - 0.5829410552978516
  - RF - 121.12735795974731
  - GB - 198.9122121334076
  - XGB - 6.422184944152832
- After Hyperparameter Tuning and feature selection (Training time in secs)
  - DT - 0.13904380798339844
  - LR - 0.19768691062927246
  - Guassian NB - 0.010295867919921875
  - RF - 11.18871283531189
  - GB - 10.678267002105713
  - XGB - 18.770753860473633