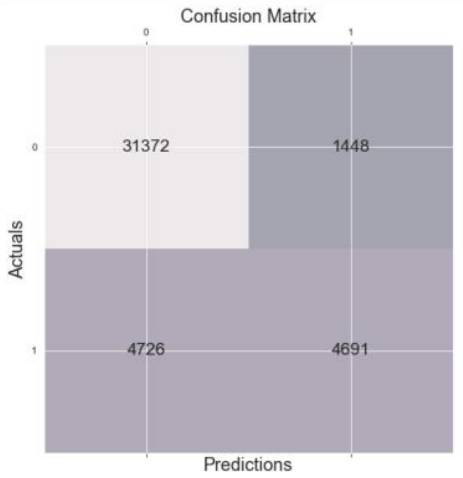
Project Development Phase
Model Performance Test

Date	18 November 2022
Team ID	**PNT2022TMID34754**
Project Name	Exploratory Analysis of Rainfall Data in India for Agriculture
Maximum Marks	10 Marks

Model Performance Testing:

S.No.	Parameter	Values	Screenshot

1.	Metrics	**Classification Model: Random Forest**
		
		Confusion Matrix –
		[[31372 1448]
		[4726 4691]]
		
		Accuracy Score-
		0.8538248455145963
		
		Classification Report –
		Accuracy:
		0.8538248455145963
		Precision:
		0.7641309659553673
		Recall:
		0.49814165870234683
		F1-score:
		0.6031113396760092
			Random forest Confusion matrix
			`conf_matrix = metrics.confusion_matrix(y_test,t1)`
			`fig,ax = plt.subplots(figsize=(7.5,7.5))`
			`ax.matshow(conf_matrix,alpha=0.3)`
			`for i in range(conf_matrix.shape[0]):`
			`  for j in range(conf_matrix.shape[1]):`
			`  ax.text(x=j, y=i, s=conf_matrix[i,j], va ='center', ha='center',size='xx-larg`
			`plt.xlabel('Predictions',fontsize=18)`
			`plt.ylabel('Actuals',fontsize=18)`
			`plt.title('Confusion Matrix',fontsize=18)`
			`plt.show()`
			 
			`t1 = Rand_forest.predict(X_test_scaled)`
			`print("Rand_forest:",metrics.accuracy_score(y_test,t1))`
			`Rand_forest: 0.8538248455145963`

			``
			print("*"*10, "Classification Report", "*"*10)
			
			print("-"*30)
			print(classification_report(y_test, t1))
			print("-"*30)
			``
			``
			********** Classification Report **********
			------------------------------ 	
			              precision recall f1-score support
			
			           0 0.87 0.96 0.91 32820
			           1 0.76 0.50 0.60 9417
			
			    accuracy 0.85 42237
			   macro avg 0.82 0.73 0.76 42237
			weighted avg 0.85 0.85 0.84 42237
			
			------------------------------ 	
			``

2.	Tune the Model	Hyperparameter Tuning & Validation Method - RandomizedSearchCV	**Hyperparameter Tuning**
			``
			from sklearn.ensemble import RandomForestRegressor
			rf = RandomForestRegressor(random_state = 42)
			from pprint import pprint
			# Look at parameters used by our current forest
			print('Parameters currently in use:\n')
			pprint(rf.get_params())
			``
			``
			Parameters currently in use:
			
			{'bootstrap': True,
			 'ccp_alpha': 0.0,
			 'criterion': 'mse',
			 'max_depth': None,
			 'max_features': 'auto',
			 'max_leaf_nodes': None,
			 'max_samples': None,
			 'min_impurity_decrease': 0.0,
			 'min_impurity_split': None,
			 'min_samples_leaf': 1,
			 'min_samples_split': 2,
			 'min_weight_fraction_leaf': 0.0,
			 'n_estimators': 100,
			 'n_jobs': None,
			 'oob_score': False,
			 'random_state': 42,
			 'verbose': 0,
			 'warm_start': False}
			``

```
n_estimators = [10,20,30,50]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 50, num = 8)]
min_samples_split = [4, 8, 10]
min_samples_leaf = [2, 4, 6]
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

```
from sklearn.model_selection import RandomizedSearchCV
rf = RandomForestRegressor()
rf_random = RandomizedSearchCV(estimator = rf,param_distributions = random_grid,r
```

```
rf_random.fit(X_train_scaled, y_train)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits

RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [10, 15, 21, 27, 32, 38,
                                                      44, 50],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [2, 4, 6],
                                        'min_samples_split': [4, 8, 10],
                                        'n_estimators': [10, 20, 30, 50]},
                   random_state=35, verbose=2)
```

```
best_params = rf_random.best_params_
```

```
print ('Best Parameters is', best_params)
```

```
Best Parameters is {'n_estimators': 50, 'min_samples_split': 10, 'min_samples_l
eaf': 6, 'max_features': 'sqrt', 'max_depth': 21, 'bootstrap': False}
```

```
print(f'Accuracy =: {round(rf_random.score(X_train_scaled, y_train) * 100, 2)}%')
```

```
Accuracy =: 75.07%
```