shaadi.com

Case Solution Approach and Results

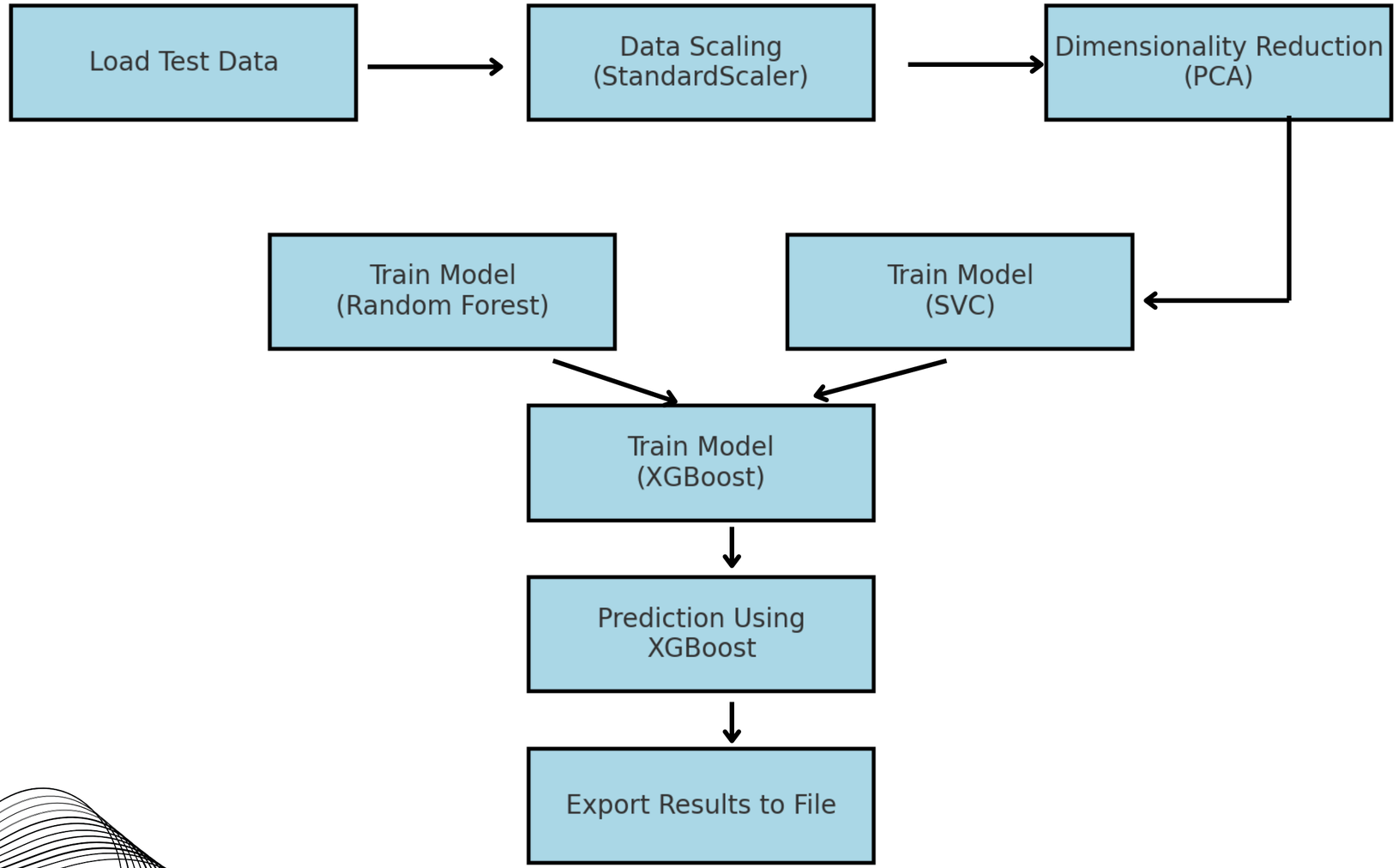
PRASHANT MATHUR

Problem Statement

•**Dataset.txt** contains customer's records with variable 'C' specifying whether the customers bought a specific product or not. F1-F22 are features, largely demographics features and also some features around customers past activities. Some of the features may contain aggregated data on the customer, including data of buying the same product in the past.

•File **'Dataset_test.txt'** to be used for testing and doesn't contain the variable 'C' which needs to be predicted.

FLOW


Load Test Data
Data Scaling
(StandardScaler)
Dimensionality Reduction
(PCA)
Train Model
(Random Forest)
Train Model
(SVC)
Train Model
(XGBoost)
Prediction Using
XGBoost
Export Results to File

APPROACH OVERVIEW

- **Data Preprocessing:** Applied scaling techniques to standardize data.
- **Dimensionality Reduction**: Principal Component Analysis (PCA) to reduce data complexity.
- **Model Training**: Trained three models - Random Forest, SVC, and XGBoost - to compare performance.
- **Test Prediction**: Used XGBoost for generating predictions on the test dataset.
- **Result Export:** Exported the predictions in a structured format.

DATA PREPROCESSING

- StandardScaler was used to normalize the dataset, ensuring uniform feature scaling.
- This step was critical to optimize the PCA and model performance.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
scaler = StandardScaler()
scaler.fit(X)
scaled_data = scaler.transform(X)
pca = PCA(n_components=5)
pca.fit(scaled_data)
x_pca = pca.transform(scaled_data)
```

[7] ✓ 0.6s

DIMENSIONALITY REDUCTION

- PCA was applied to reduce the dimensionality of the dataset to 5 principal components.
- This retained the most significant information while minimizing computation overhead

```
df_new = pd.DataFrame(data = x_pca, columns = ['PC1', 'PC2','PC3','PC4','PC5'])
df_new['C'] = y
```

```
df_new.head()
```

	PC1	PC2	PC3	PC4	PC5	C
0	1.434548	0.584591	-0.223435	-0.043214	-0.102274	0
1	0.429526	-0.695746	-0.424244	-1.091889	-1.603673	1
2	1.467669	0.672749	-2.240887	0.232094	1.274789	0
3	-0.431834	-1.119765	0.227681	0.207635	-1.928224	0
4	0.486209	-0.520013	-1.257975	-1.610753	0.298727	0

BALANCING DATA WITH UNDERSAMPLING

- Balanced the dataset by reducing the majority class using RandomUnderSampler for better model performance.

- Ensured equal class distribution by resampling features and target, verified through class distribution statistics

```python
from imblearn.under_sampling import RandomUnderSampler

# Initialize the RandomUnderSampler with stratification
undersampler = RandomUnderSampler(sampling_strategy='auto', random_state=42)

# Apply undersampling
X_resampled, y_resampled = undersampler.fit_resample(X, y)

# Combine resampled features and target into a DataFrame
undersampled_data = pd.concat([pd.DataFrame(X_resampled, columns=X.columns), pd.Series(y_resampled, name='C')], axis=1)

# Print class distribution after undersampling
print("Class distribution after stratified undersampling:\n", undersampled_data['C'].value_counts())
```

✓ 0.1s Python

```
Class distribution after stratified undersampling:
 0    24827
1    24827
```

MODEL TRAINING AND PREDICTION

- Three models were trained: Random Forest, SVC, and XGBoost.
- Each model was evaluated, but XGBoost demonstrated the best performance for predictions on the test dataset.
- Predictions were generated using the trained XGBoost model.
- The results were exported for further analysis.

- XGBoost

```
]  ✓ 0.0s
accuracy:0.682307924680294
precision:0.6229530383001759
recall:0.9257843925985519
f1_score:0.7447617506674218
```

- SVC

```
accuracy:0.6773738797704159
precision:0.6217295510878545
recall:0.9080852775543041
f1_score:0.7381069151544875
```

- Random Forest

```
accuracy:0.6552210250730037
precision:0.6206359102244389
recall:0.8008849557522124
f1_score:0.6993326308394802
```

RESULTS AND INSIGHTS

- The model demonstrated reasonable predictive performance on the test dataset.
- Principal components effectively captured the variance in the data.
- The output was structured and ready for integration into decision systems.

CONCLUSION

The approach successfully utilized data preprocessing, dimensionality reduction, and machine learning to achieve the desired predictions. This scalable methodology can be adapted for similar challenges in the future.