AGENT PERFORMANCE CLASSIFICATION &

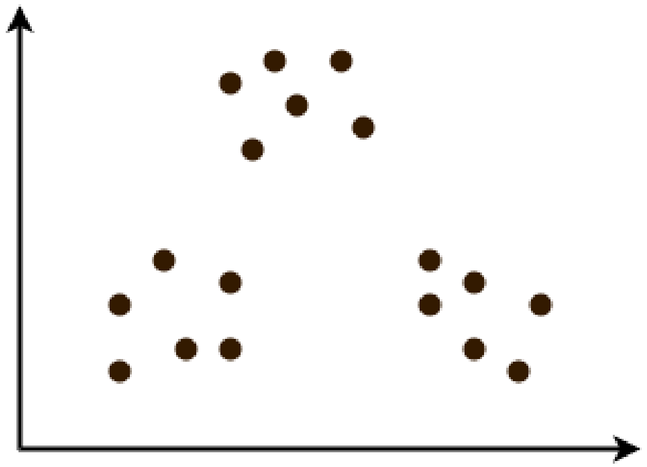
PERSONALIZED INTERVENTION STRATEGY RECOMMENDATION

HELLOWORLD 2.0

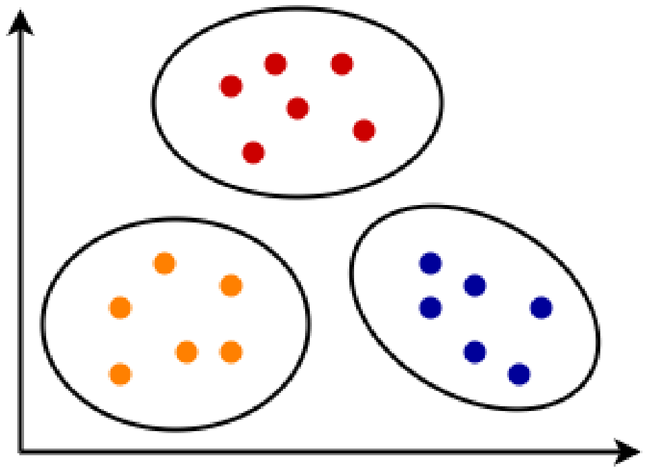
Performance Features

1. new_policy_count
2. ANBP_value
3. net_income
4. avg_policy_value
5. profit_per_policy
6. overall_conversion_rate
7. proposal_to_quotation_rate
8. quotation_to_policy_rate
9. unique_proposals_last_21_days
10. activity_rate_21days
11. unique_customers

Classification Strategy



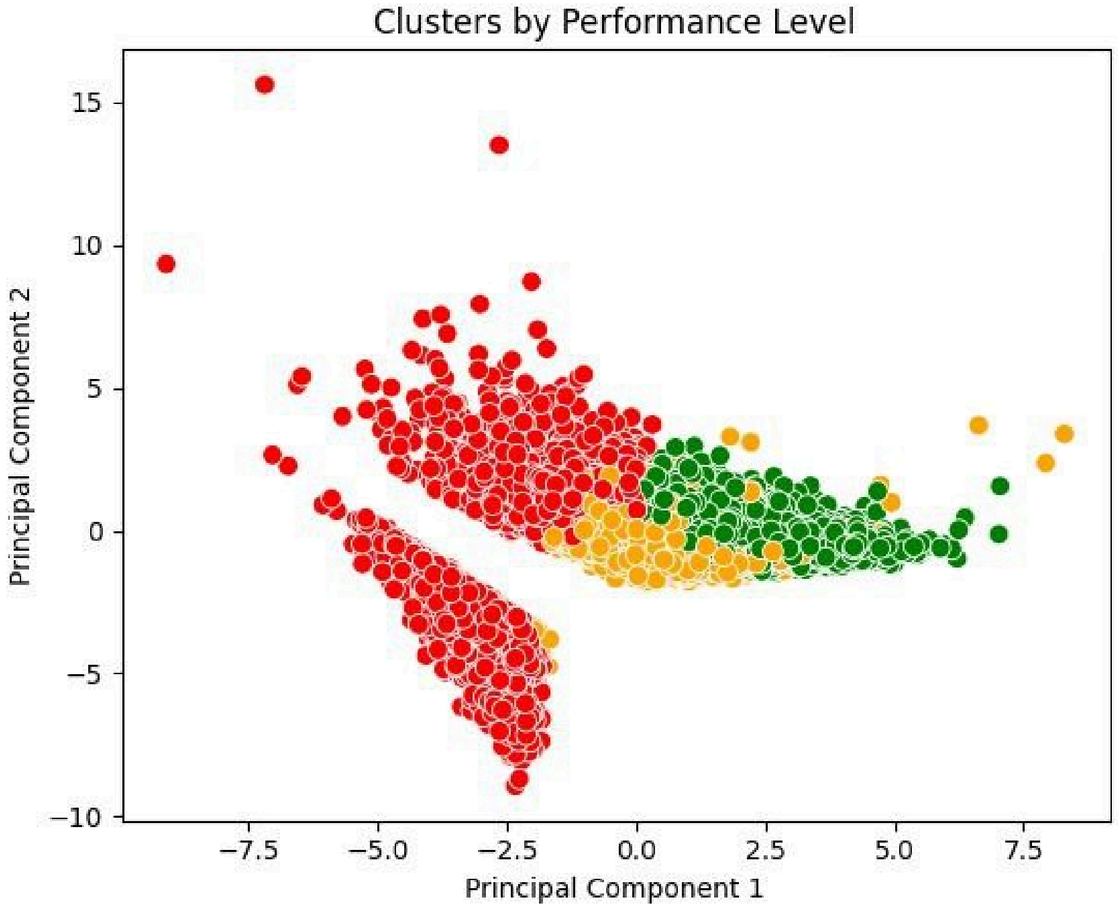
Before K-Means



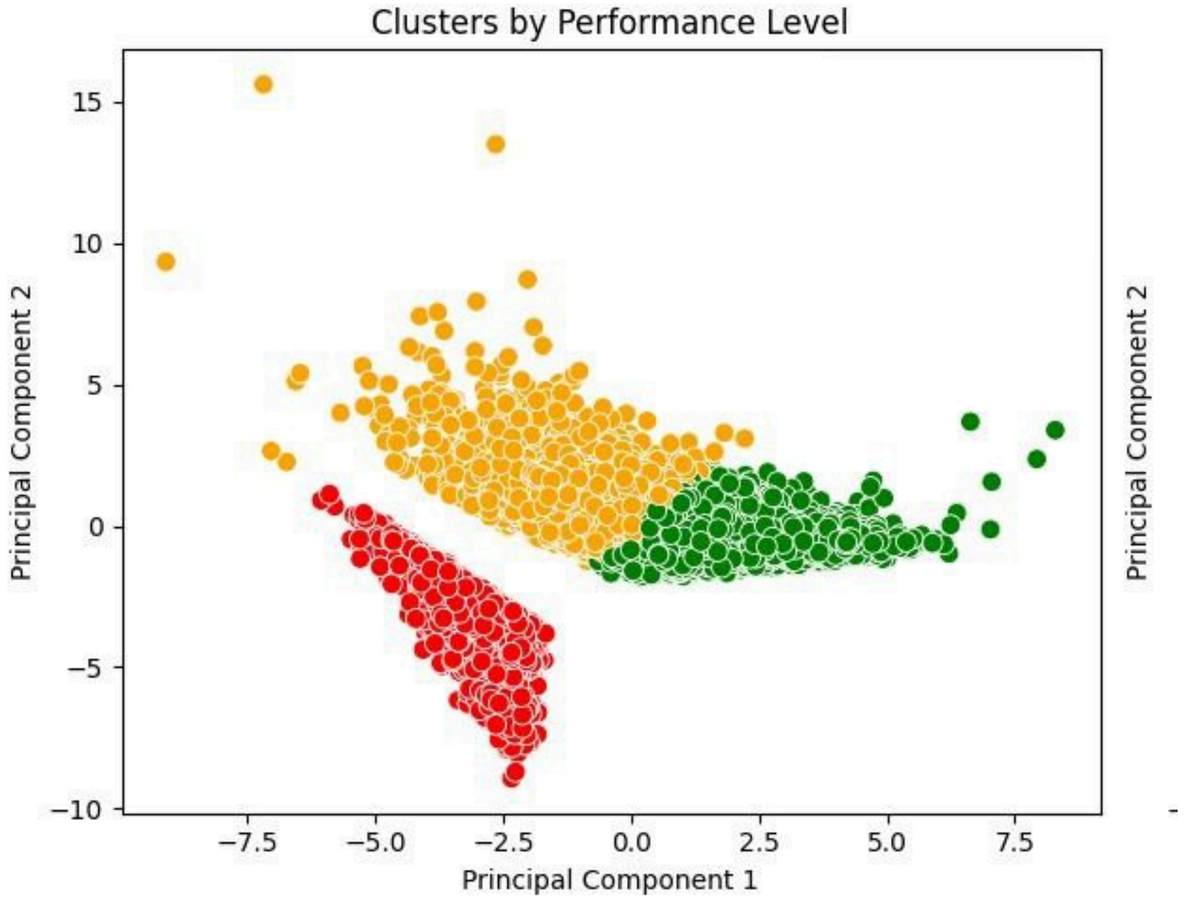
After K-Means

K-means Clustering

K-means Clustering - PCA Analysis



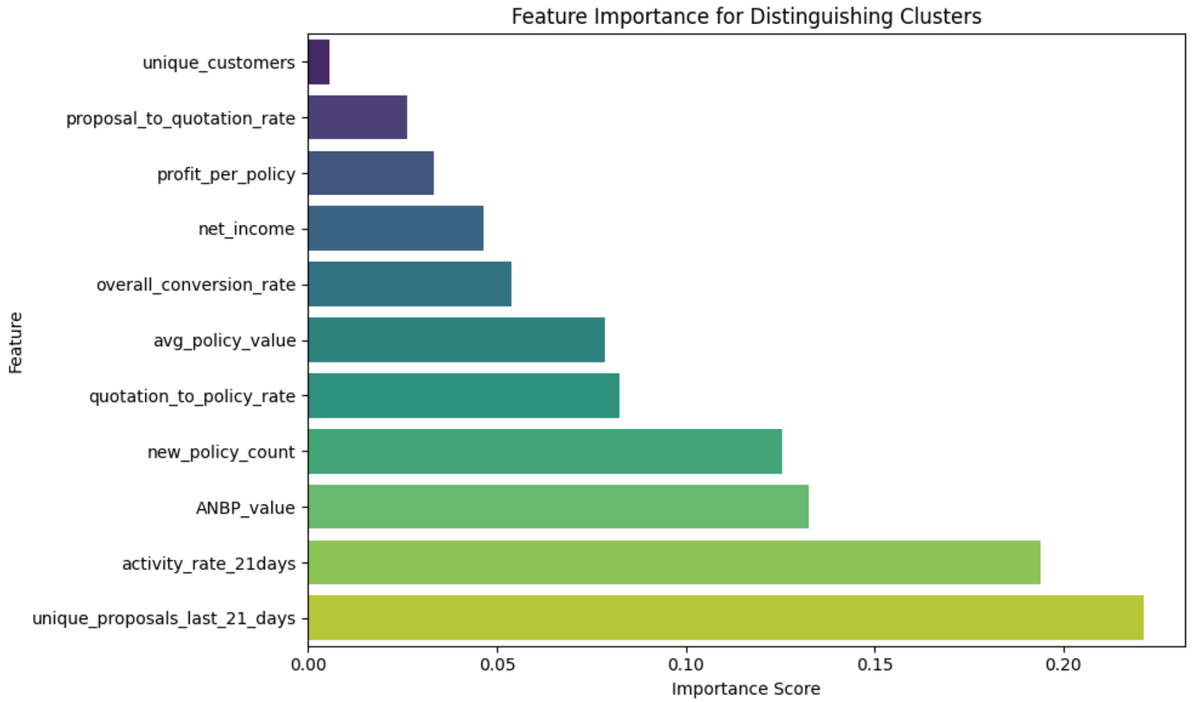
Random State =10



Random State =42

PCA Analysis

Calculate a composite score for each cluster




```
composite_scores = (
    cluster_centers['unique_proposals_last_21_days'] * 0.221 +
    cluster_centers['activity_rate_21days'] * 0.194 +
    cluster_centers['ANBP_value'] * 0.133 +
    cluster_centers['new_policy_count'] * 0.126 +
    cluster_centers['quotation_to_policy_rate'] * 0.082 +
    cluster_centers['avg_policy_value'] * 0.078 +
    cluster_centers['overall_conversion_rate'] * 0.054 +
    cluster_centers['net_income'] * 0.046 +
    cluster_centers['profit_per_policy'] * 0.033 +
    cluster_centers['proposal_to_quotation_rate'] * 0.026 +
    cluster_centers['unique_customers'] * 0.006
)
```

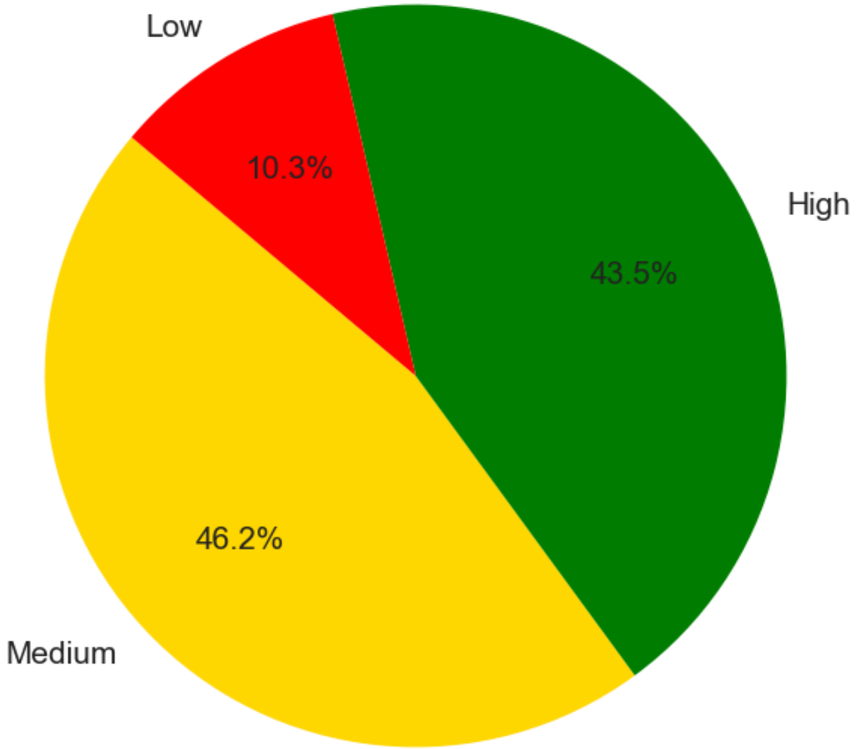
Giving weighted importance to each feature

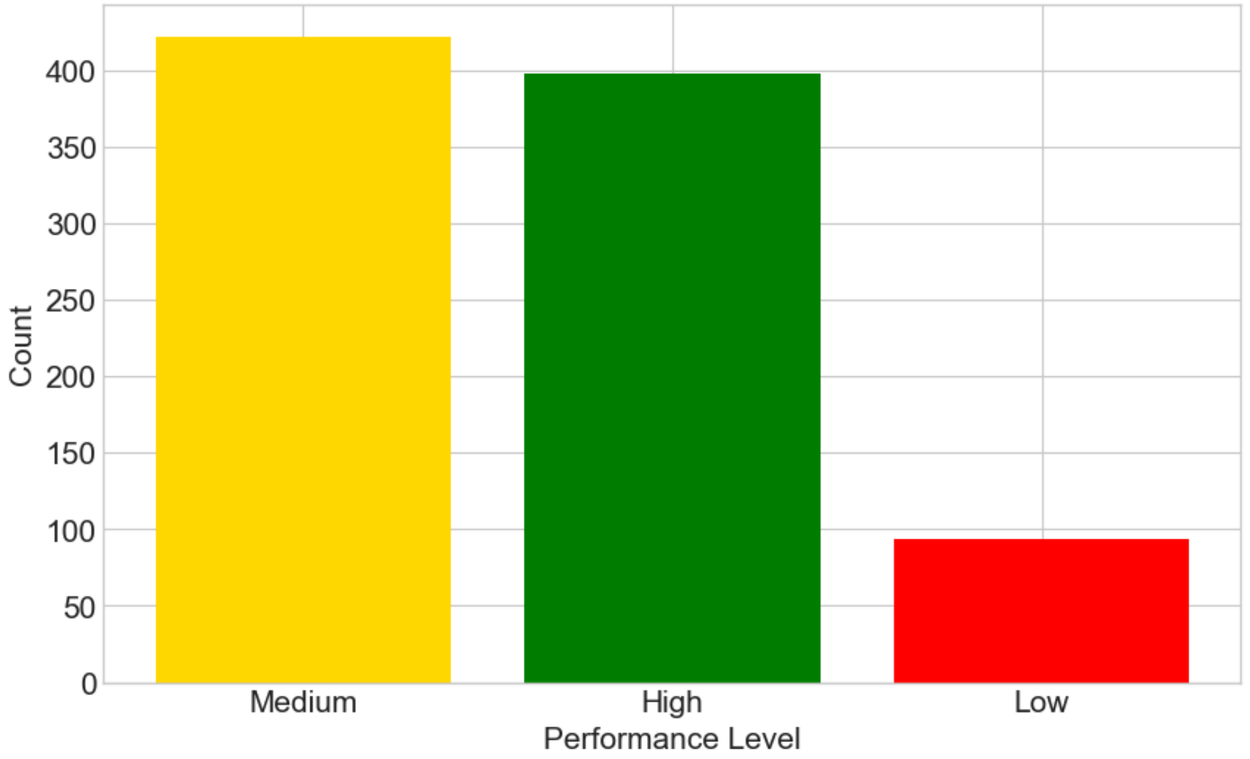
Rank clusters based on composite score

```
cluster_rankings = composite_scores.rank(ascending=False)
performance_mapping = {}

for cluster in range(3):
    if cluster_rankings[cluster] == 1:
        performance_mapping[cluster] = 'High'
    elif cluster_rankings[cluster] == 2:
        performance_mapping[cluster] = 'Medium'
    else:
        performance_mapping[cluster] = 'Low'
```

Performance Level Distribution





- High - 398
- Medium - 422
- Low - 94