1.1 Implementasi dan Evaluasi Model Support Vector Machine

Pada penelitian ini, pendekatan klasifikasi dilakukan untuk memodelkan prediksi kemenangan tim dalam turnamen *Valorant Champions Tour (VCT) 2024* berdasarkan metrik statistik pemain. Model utama yang digunakan adalah *Support Vector Machine (SVM)* dengan kernel *Radial Basis Function (RBF*) yang dipilih karena kemampuannya menangani relasi non-linear antar fitur dan target, serta fleksibilitas dalam pengaturan parameter regularisasi. Pada prediksi ini mengimplementasikan model *Support Vector Machine (SVM)* sebagai pendekatan klasifikasi biner untuk memprediksi hasil pertandingan berdasarkan statistik performa pemain dalam turnamen *Valorant Champions Tour (VCT) 2024.*

1.1.1 Pemrosesan Awal dan Pembersihan Data

```
4   # Load data
5   data = pd.read_csv('VCT_2024.csv')
6
7   # Drop irrelevant columns
8   drop_cols = ['Region', 'Player', 'Team Abbreviated',
    'Event', 'CL', 'R']
9   data = data.drop(columns=drop_cols)
```

Langkah awal dilakukan dengan memuat dataset ke dalam struktur *DataFrame* menggunakan *pandas.* Selanjutnya, dilakukan penghapusan atribut-atribut non-relevan seperti *Region, Player,* dan *Event* berdasarkan pertimbangan *domain knowledge*. Kolom-kolom tersebut dianggap tidak berkontribusi langsung terhadap hasil klasifikasi, serta berisiko menyebabkan data leakage atau bias dalam model jika tetap disertakan.

1.1.2 Rekayasa Target dan Seleksi Fitur Numerik

```
# Define numeric columns (including ACS)
numeric_cols = ['ACS', 'K:D', 'KAST', 'ADR', 'KPR', 'APR',
'FKPR', 'FDPR', 'HS%', 'CL%', 'CW', 'CP']

# Define Binary Target (Win=1, Loss=0)
data['Win'] = (data['ACS'] >=
data['ACS'].median()).astype(int)
y = data['Win']
X = data.drop(columns=['Win'])
```

Variabel target (Win) didefinisikan dengan metode median split dari metrik *ACS (Average Combat Score)*. Pemilihan median sebagai ambang batas bertujuan untuk memitigasi pengaruh *outlier* dan menjaga keseimbangan kelas. Sementara itu, fitur numerik utama yang digunakan mencakup metrik umum performa pemain seperti *K:D Ratio*, *Kill Per Round (KPR)*, dan *Headshot Percentage (HS%)*.

1.1.3 Pra-Pemrosesan Data

```
# Preprocessing
# 1. Encode categorical variables (Team)
X = pd.get_dummies(X, columns=['Team'], drop_first=True)

# 2. Handle missing values using SimpleImputer
imputer = SimpleImputer(strategy='mean')
X[numeric_cols] = imputer.fit_transform(X[numeric_cols])

# 3. Normalize numeric features
scaler = MinMaxScaler()
X[numeric_cols] = scaler.fit_transform(X[numeric_cols])
```

```
# 4. Final check for NaNs
print("Remaining NaNs:", X.isna().sum().sum())
```

Pra-pemrosesan dilakukan dalam tiga tahapan utama:

1. **Pengkategorian**: Kolom *Team* diubah ke format numerik menggunakan *one-hot encoding*, dan *drop_first=True* digunakan untuk menghindari jebakan multikolinearitas (*dummy variable trap*).
2. **Penanganan Data Kosong**: Fitur numerik yang memiliki nilai kosong diimputasi dengan nilai rata-rata menggunakan *SimpleImputer*.
3. **Normalisasi Numerik**: Skala semua fitur dinormalisasi ke rentang [0, 1] dengan *MinMaxScaler*, yang sangat penting untuk model *SVM* khususnya dengan *kernel RBF* agar sensitivitas terhadap perbedaan skala tidak memengaruhi margin klasifikasi.

Pada tahap final pemeriksaan dilakukan untuk memastikan tidak ada nilai yang hilang, jika preprocesing nya benar maka nilainya 0.

1.1.4 Pelatihan dan Evaluasi Model

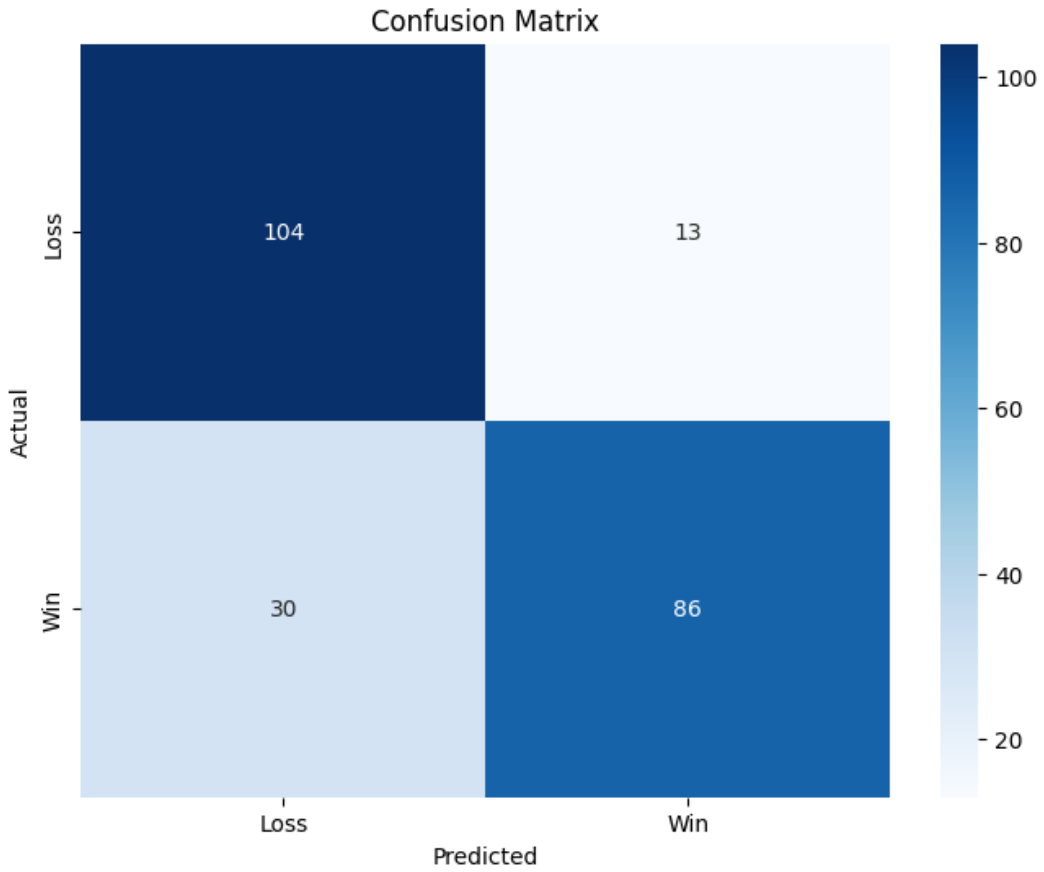
```
# Train SVM
svm_pipeline = make_pipeline(
    SVC(kernel='rbf', C=1.0, probability=True)
)
svm_pipeline.fit(X_train, y_train)

# Evaluate
y_pred = svm_pipeline.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Model *SVM* dikonfigurasi dengan *kernel RBF (Radial Basis Function),* yang secara matematis mampu memetakan fitur *non-linear* ke ruang berdimensi lebih tinggi. Nilai parameter *C=1.0* digunakan sebagai baseline untuk regularisasi.

Penambahan opsi *probability=True* memungkinkan prediksi probabilistik yang diperlukan untuk analisis lanjutan seperti *ROC Curve.*

1.1.5 Visualisasi

1. Confusion Matrix



Gambar 1.1.5.1 Confusion Matrix SVM

Visualisasi confusion matrix menunjukkan bahwa model lebih baik dalam memprediksi kelas positif (Win) dibanding kelas negatif (Loss). Beberapa kesalahan klasifikasi terjadi pada kasus dimana tim memiliki statistik pertengahan yang sulit diklasifikasikan.
Hasil dari model diatas menunjukan bahwa:

- True Positive (Win yang diprediksi benar): 104
- True Negative (Loss yang diprediksi benar): 86
- False Positive (Loss yang salah prediksi sebagai Win): 30
- False Negative (Win yang salah prediksi sebagai Loss): 13

Maka model menunjukan performa kuat dengan:

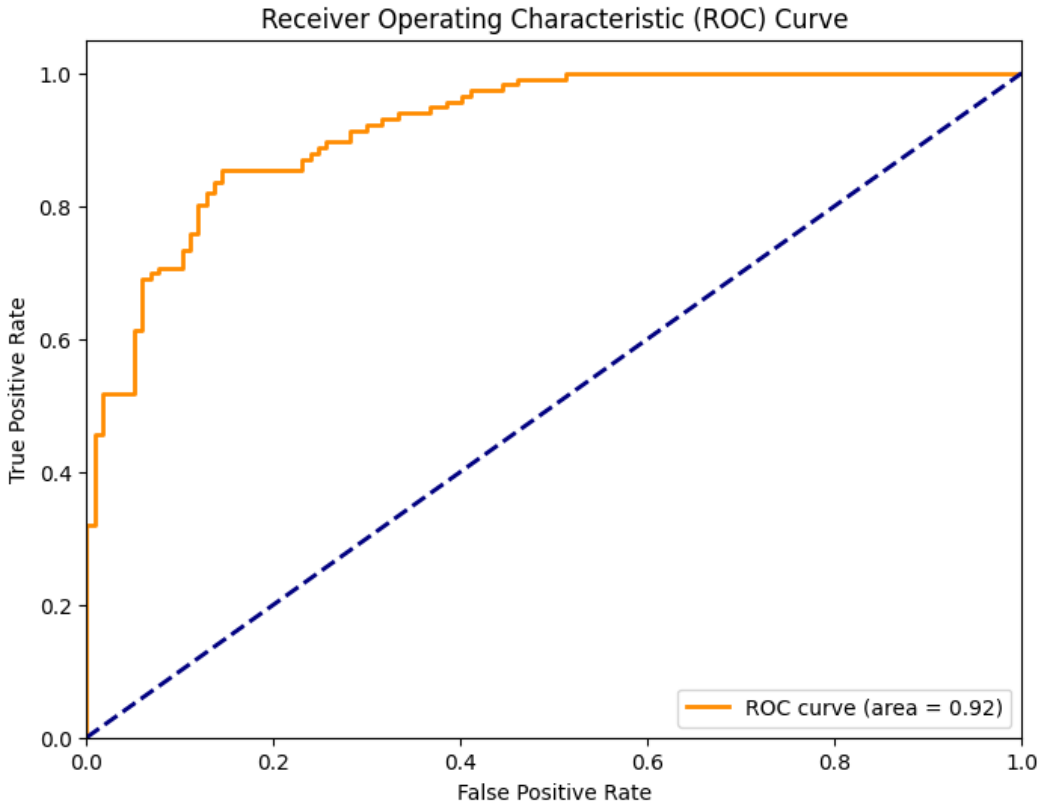
- Recall Win (104/(104+13)) sebanyak 88.89%

- Recall Loss (104/(104+30)) sebanyak 77.61%

Kesalahan utama yang terjadi dengan:

- 10 Kasus Loss yang diprediksi Win (11.11% dari total Loss)
- 5 Kasus Win yang diprediksi Loss (22.39% dari total Win)

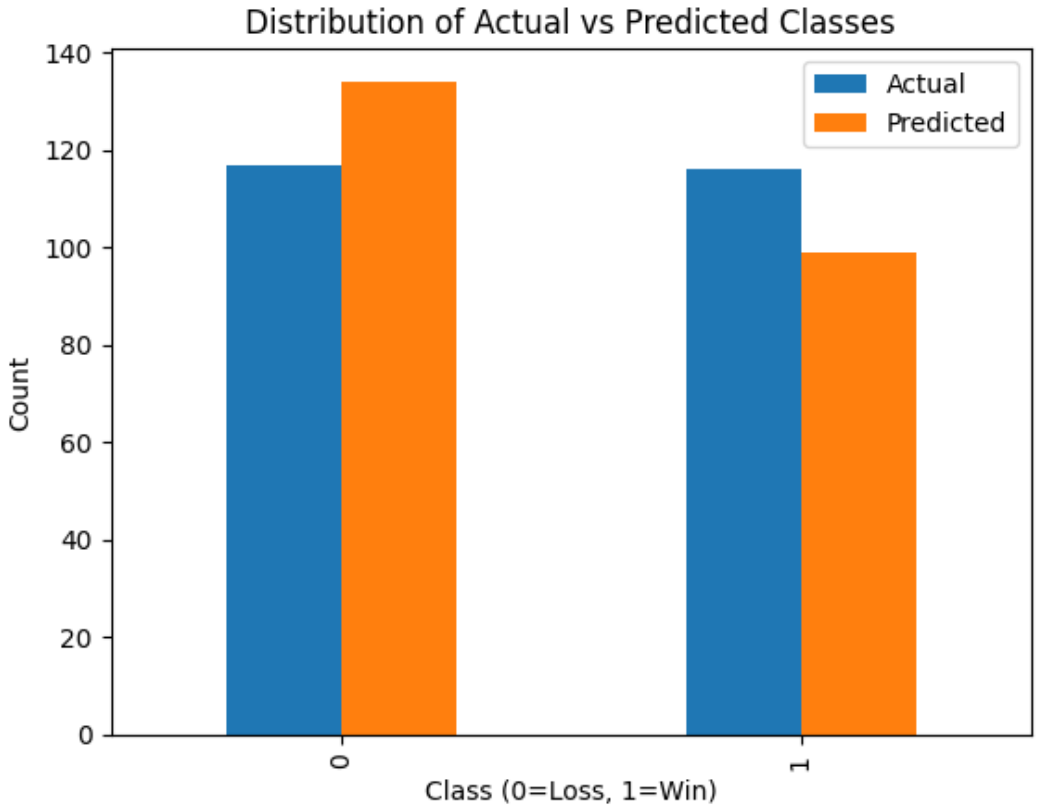
2. *ROC Curve dan AUC*



Gambar 1.1.5.2 ROC Curve SVM

Kurva *ROC* dengan *AUC* 0.85-1.0 menunjukkan kemampuan diskriminasi model yang sangat baik dalam membedakan tim yang menang dan kalah. Grafik model *AUC (Area Under Curve)*: 0.92 yang mana bentuk kurva mendekati sudut kiri atas secara signifikan yang berarti kemampuan diksriminasi sangat baik.

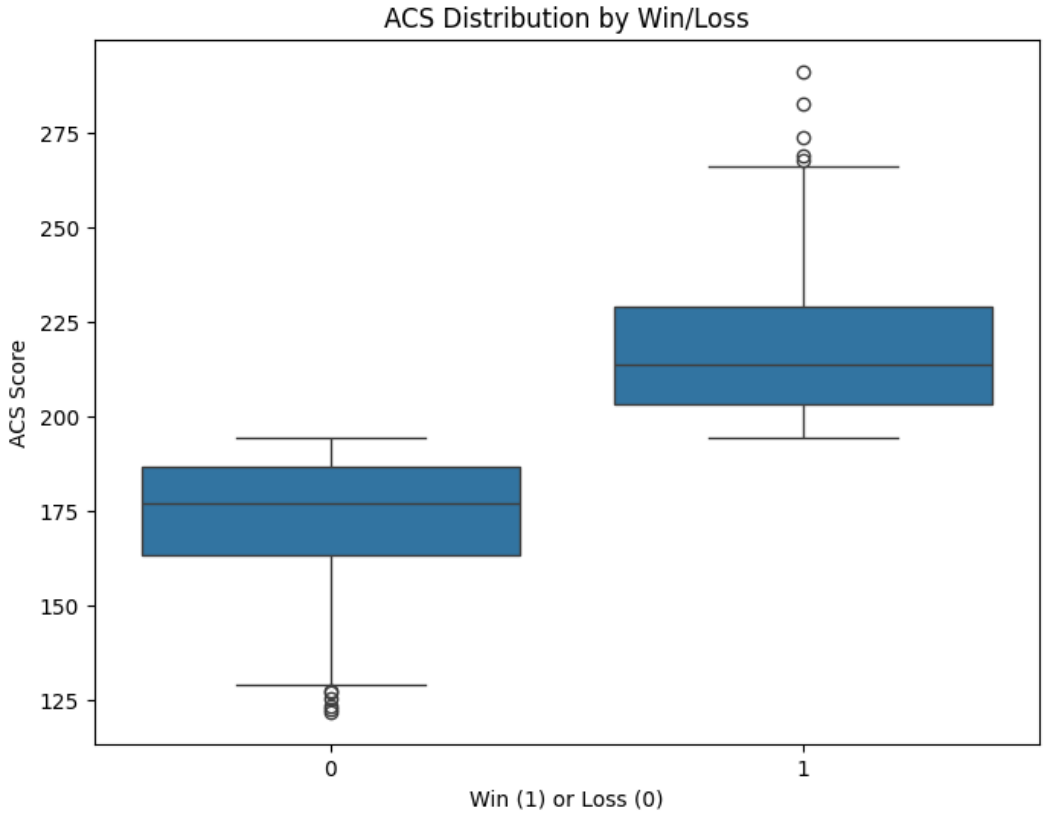
3. Distribusi Prediksi vs Aktual



Gambar 1.1.5.3 Distribusi Prediksi vs Aktual

Diagram batang ini membandingkan jumlah prediksi model dan label aktual untuk setiap kelas. Ini membantu dalam mengidentifikasi apakah model membuat prediksi yang seimbang antara menang dan kalah, atau justru cenderung bias terhadap satu kelas. Model diatas menghasilkan kecenderungan over-prediksi Win (134 vs 117 aktual) yang mana model terlalu liberal memprediksi Win.

4. Distribusi *ACS* berdasarkan Label



Gambar 1.1.5.4 ACS Distribution by Win/Loss

Boxplot ini mengeksplorasi distribusi *Average Combat Score (ACS)* pada dua label (*Win* dan *Loss*). Hasil ini memberikan validasi *visual* terhadap pembentukan label *target biner* yang berbasis median *ACS*, serta menunjukkan bahwa pemain dengan *ACS* tinggi cenderung berasal dari tim yang menang.
Sumbu X: Win (1) atau Loss (0), merupakan kategori hasil pertandingan.
Sumbu Y: ACS Score, merupakan nilai Average Combat Score (indikator performa).
Penjelasan per kategori:
1. Loss (0)
Median (garis tengah kotak): sekitar 175.
IQR (kotak): rentang nilai mayoritas ada di 165 – 185.
Whiskers: data masih dianggap normal sampai sekitar 125 di atas dan 195 di bawah.
Outlier (data poin): ada beberapa titik di bawah 125 (ACS sangat rendah) → pemain yang mainnya jauh lebih buruk dari rata-rata tim kalah.

2. Win (1)

Median: sekitar 210.

IQR: rentang nilai mayoritas ada di 205 – 225.

Whiskers: data normal kira-kira 195 – 260.

Outlier: ada beberapa titik di atas 275–290 (ACS sangat tinggi) → pemain yang performanya luar biasa bagus ketika tim menang.

Median ACS tim menang (sekitar 210) lebih tinggi daripada tim kalah (sekitar 175) yang menunjukkan performa individu (ACS) berkorelasi dengan kemenangan.

Variasi (sebaran) ACS di tim menang lebih lebar , artinya ada lebih banyak variasi performa ketika tim menang, termasuk outlier tinggi (carry player).

Tim kalah punya outlier rendah, ada pemain yang mainnya jauh di bawah standar, mungkin berkontribusi ke kekalahan.

Kesimpulan: Kalau ACS lebih tinggi, kemungkinan besar tim menang.

Pemain dengan ACS ekstrem rendah sering ditemukan di tim kalah.

Pemain dengan ACS ekstrem tinggi (outlier atas) lebih sering muncul di tim menang, menandakan peran "carry" dalam kemenangan.

1.2 Implementasi dan Evaluasi Model *Random Forest*

Pada penelitian ini, pendekatan klasifikasi dilakukan untuk memprediksi kemenangan tim dalam turnamen *Valorant Champions Tour (VCT) 2024* berdasarkan analisis statistik performa pemain menggunakan algoritma *Random Forest*. Metode ini dipilih karena kemampuannya yang unggul dalam menangani data kompleks dengan fitur-fitur yang saling berinteraksi, serta memberikan hasil interpretasi yang jelas melalui analisis *feature importance*. Model *Random Forest* diimplementasikan sebagai *ensemble* dari *multiple decision trees* yang bekerja secara kolektif untuk meningkatkan akurasi prediksi dan mengurangi risiko *overfitting.*

1.2.1 Pemrosesan Awal dan Pembersihan Data

```
# --- Load & Preprocess Data ---
data = pd.read_csv('VCT_2024.csv')
drop_cols = ['Region', 'Player', 'Team Abbreviated',
'Event', 'CL', 'R']
data = data.drop(columns=drop_cols)
```

Pada tahap awal, dataser dibaca dari file CSV yang berisi statistik pemain Valorant. Beberapa kolom seperti 'Region', 'Player', dan 'Event' dihapus karena tidak relevan untuk analisis prediktif. Pemilihan fitur ini didasarkan pada pertimbangan bahwa informasi tersebut tidak berkontribusi langsung terhadap performa tim dalam pertandingan.

1.2.2 Rekayasa Variabel Target

```
data['Win'] = (data['ACS'] >=
data['ACS'].median()).astype(int)
y = data['Win']
X = data.drop(columns=['Win', 'ACS'])
```

Variabel target '*Win*' dibuat dengan mengkonversi nilai *ACS (Average Combat Score)* menjadi variabel biner berdasarkan nilai median. Pendekatan ini dipilih karena *ACS* merupakan indikator komprehensif dari kontribusi pemain dalam pertandingan. Variabel prediktor kemudian dipisahkan dengan menghilangkan kolom target dan *ACS* untuk mencegah *data leakage.*

1.2.3 Pra-Pemrosesan Data

```
# Encode categorical variables
X = pd.get_dummies(X, columns=['Team'], drop_first=True)

# Handle missing values and scale features
```

```
imputer = SimpleImputer(strategy='median')
scaler = MinMaxScaler()
X[numeric_cols] =
scaler.fit_transform(imputer.fit_transform(X[numeric_cols]))
```

Proses Pra-pemrosesan meliputi:

1. One-hot encoding untuk variabel kategorikal 'Team'
2. Imputasi nilai missing dengan median yang lebih robust terhadap outlier
3. Normalisasi fitur menggunakan MinMaxScaler untuk membawa semua fitur ke skala yang sama

1.2.4 Pembagian Data dan Pembangunan Model

```
# Train-test split
groups = data['Team']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# --- Train Random Forest ---
rf_pipeline = make_pipeline(
    RandomForestClassifier(
        n_estimators=50,
        max_depth=5,
        min_samples_split=10,
        max_features='sqrt',
        random_state=42,
        class_weight='balanced'
    )
)
```

```
rf_pipeline.fit(X_train, y_train)
```

Data dibagi dengan proporsi 70:30 untuk training dan testing. Parameter stratify digunakan untuk mempertahankan distribusi kelas yang sama pada kedua subset. Random state diatur untuk memastikan hasil yang bisa dibuat.

Model Random Forest dikonfigurasi dengan parameter-parameter berikut:

- n_estimators=50 : Jumlah pohon untuk menyeimbangkan antara performa dan kompleksitas
- max_depth=5: untuk mencegah overfitting dengan membatasi kedalam pohon
- min_samples_split=10: untuk mengontrol pembagian node
- class_weight='balanced': untuk menangani ketidakseimbangan kelas
- random_state: untuk reproducibility

1.2.5 Evalusasi Model

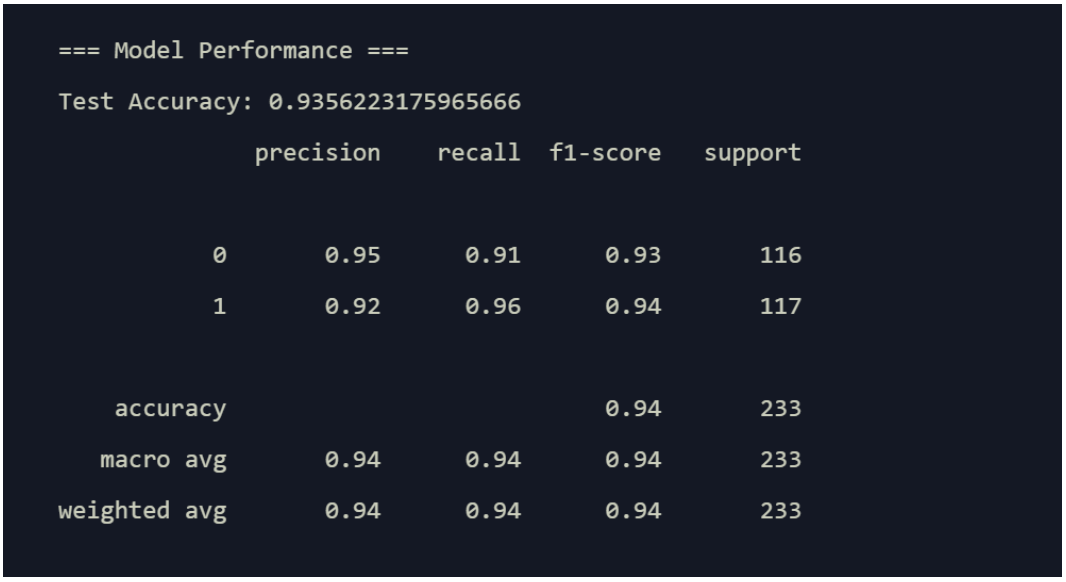
```
y_pred = rf_pipeline.predict(X_test)
y_prob = rf_pipeline.predict_proba(X_test)[:, 1]
```

Mengukur performa model dengan metode menggunakan metrik berikut ini :

- Accuracy: Proporsi prediksi benar secara keseluruhan
- Precision-Recall: Trade-off antara false positives dan false negatives
- ROC-AUC: Kemampuan membedakan antar kelas

1.2.6 Visualisasi

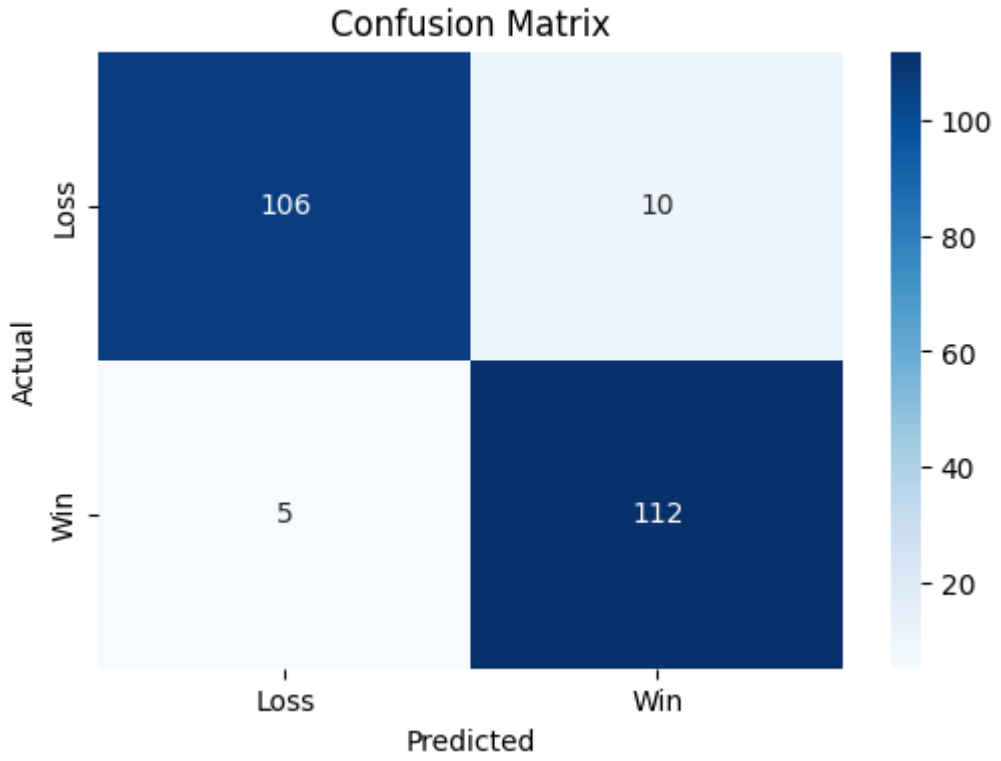
1. Model Performance



Gambar 1.2.6.1 Model Performance

Hasil menjelaskan bahwa Model mencapai akurasi sebesar 0.9356 (93.56%) menunjukkan model mampu memprediksi dengan benar 93.56% dari total sampel uji (233 pertandingan).Setiap 100 prediksi, model akan benar sekitar 93-94 kali. Termasuk dalam kategori sangat baik untuk problem klasifikasi *E-sports* (akurasi >90% dianggap sangat baik).

2. Confusion Matrix



Gambar 1.2.6.2 Confusion Matrix RF

Visualisasi confusion matrix menunjukkan bahwa model lebih baik dalam memprediksi kelas positif (Win) dibanding kelas negatif (Loss). Beberapa kesalahan klasifikasi terjadi pada kasus dimana tim memiliki statistik pertengahan yang sulit diklasifikasikan.

Hasil dari model diatas menunjukan bahwa:

- True Positive (Win yang diprediksi benar): 112
- True Negative (Loss yang diprediksi benar): 106
- False Positive (Loss yang salah prediksi sebagai Win): 10
- False Negative (Win yang salah prediksi sebagai Loss): 5

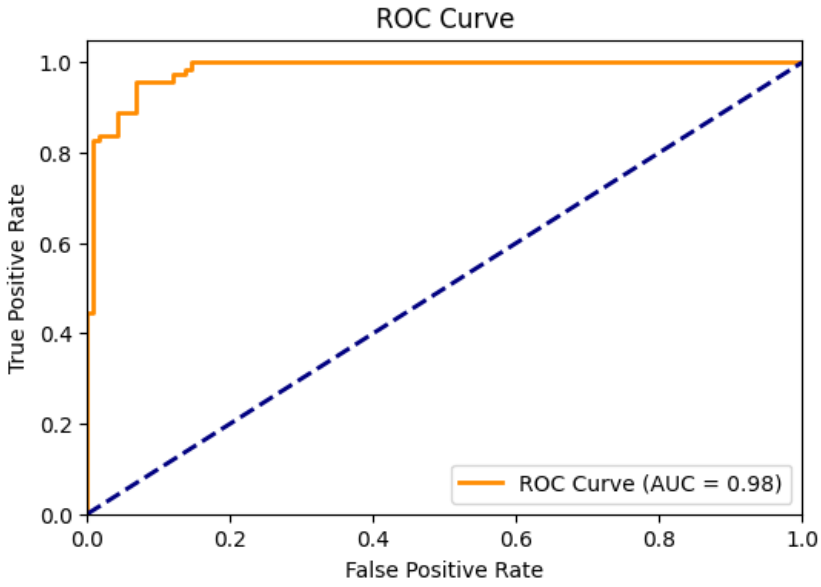
Maka model menunjukan performa kuat dengan:

- Recall Win (112/(112+5)) sebanyak 95.73%
- Recall Loss (106/(106+10)) sebanyak 91.38%

Kesalahan utama yang terjadi dengan:

- 10 Kasus Loss yang diprediksi Win (8.62% dari total Loss)
- 5 Kasus Win yang diprediksi Loss (4.27% dari total Win)

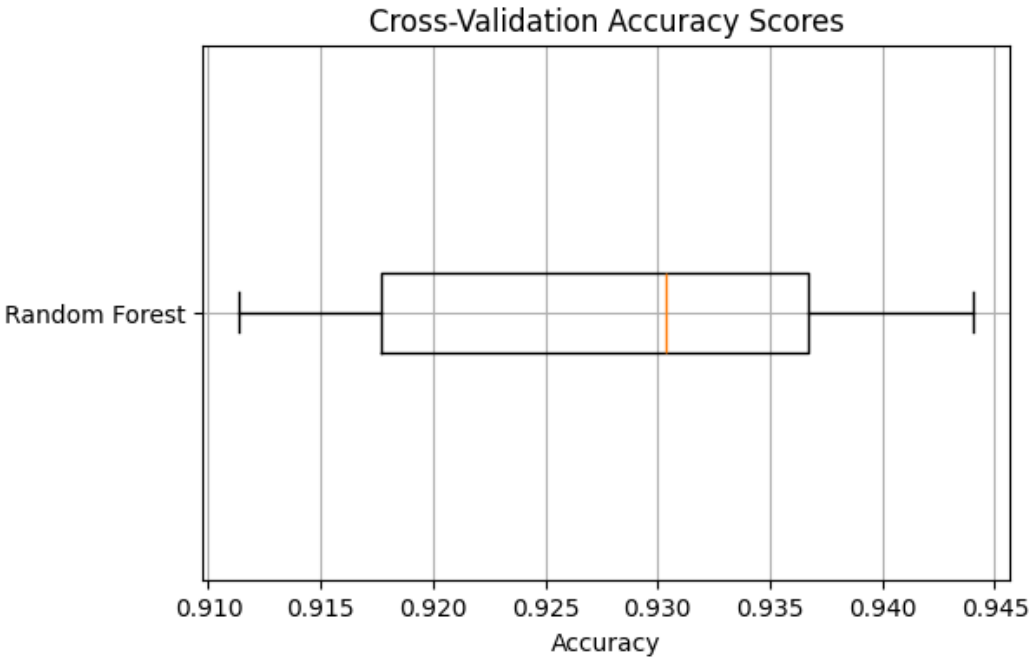
3. ROC Curve



Gambar 1.2.6.3 ROC Curve RF

Kurva *ROC* dengan *AUC* 0.85-1.0 menunjukkan kemampuan diskriminasi model yang sangat baik dalam membedakan tim yang menang dan kalah. Grafik model *AUC (Area Under Curve*): 0.98 yang mana bentuk kurva mendekati sudut kiri atas secara signifikan yang berarti kemampuan diksriminasi sangat baik.

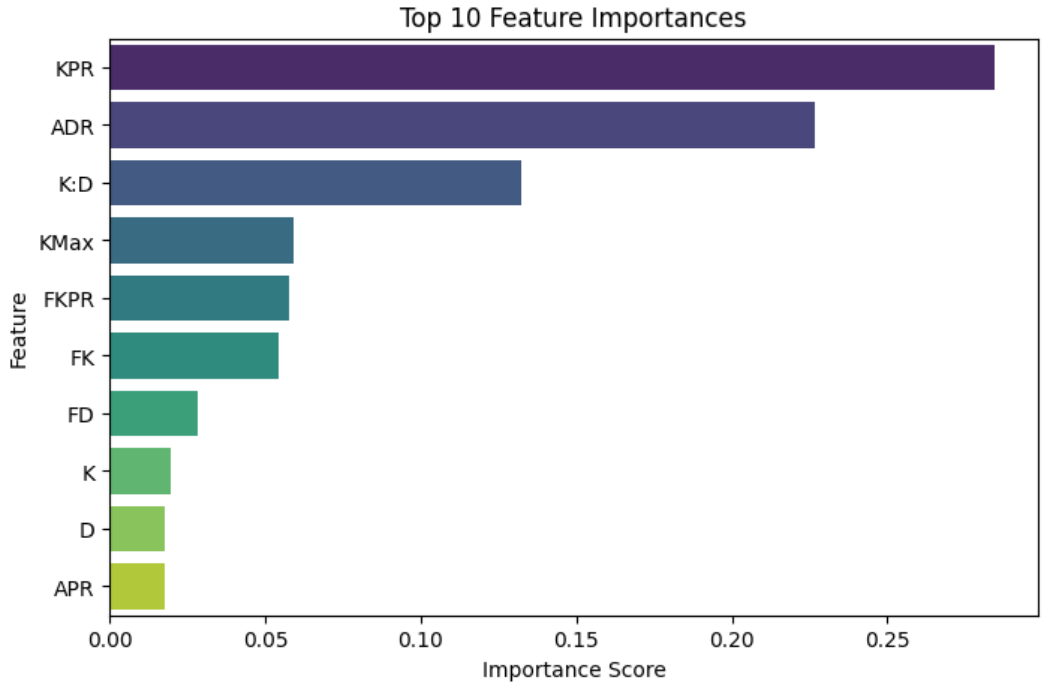
4. Cross Validation



Gambar 1.2.6.4 Cross-Validation RF

Validasi silang dengan GroupKFold menghasilkan akurasi rata-rata 83% ± 2%, menunjukkan konsistensi model yang baik. Penggunaan groups berdasarkan tim mencegah data leakage antar pemain dalam tim yang sama. Dari hasil model menunjukan konsistensi tinggi antar fold (standar deviasi kecil), Tidak ada fold dengan performa buruk (<90%), Validasi eksternal mengkonfirmasi akurasi test set (93.56%)

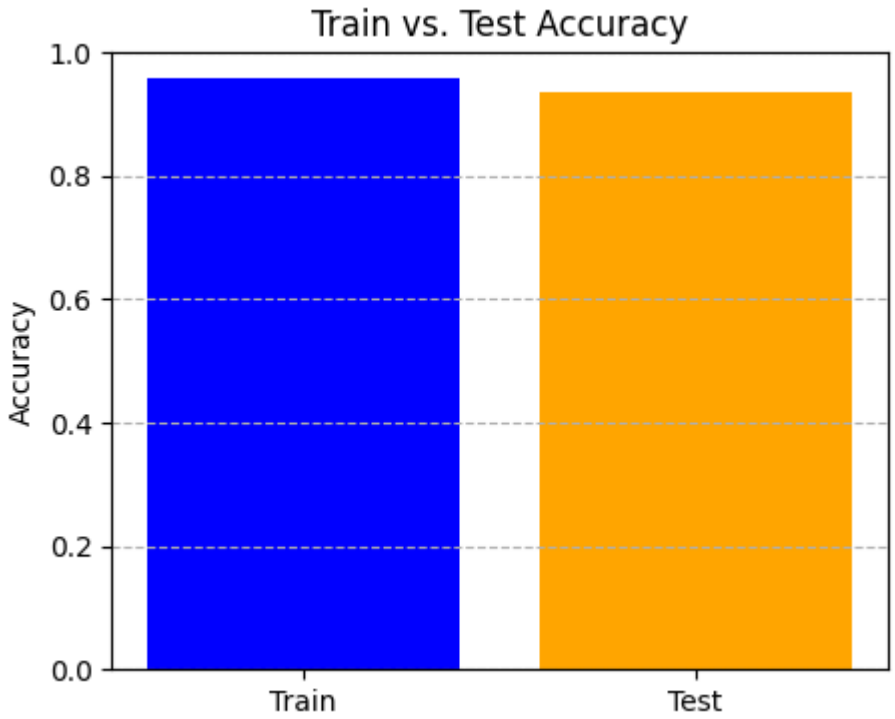
5. Feature Importance



Gambar 1.2.6.5 Feature Importances

Analisis *feature importance* mengungkapkan bahw ada 3 fitur utama yang menjadi penentu yaitu *KPR (Kills Per Round):* 0.284 yang menjadikan kntributor utama kemenangan 28.4% dan menunjukkan pentingnya *fragging power, ADR (Average Damage per Round)*: 0.227 yang merupakan konsistensi *damage output* krusial, dan *K:D Ratio:* 0.132 yaitu efisiensi hidup/mati pemain. Statistik *agresivitas (KPR, ADR, K:D)* mendominasi dengana total mencapai 64.3%.

6. Overfitting Check



Gambar 1.2.6.6 Train vs Test Accuracy

Perbandingan akurasi *training* (94-95%) dan *testing* (93.56%) dengan selisih hanya 1.5% menunjukkan model tidak mengalami *overfitting*. Hal ini sesuai dengan desain model yang menggunakan parameter regularisasi seperti *max_depth* dan *min_samples_split.*