BAB 2
LANDASAN TEORI

2.1 Penelitian Pendahulu

Penelitian ini didasarkan pada tiga penelitian yang sudah ada sebelumnya mengenai penerapan algoritma pembelajaran mesin dalam memprediksi *stunting* dan malnutrisi pada balita. Penelitian terdahulu yang menjadi dasar dari penelitian ini dapat dilihat pada Tabel 2.1

Tabel 2.1. Penelitian Pendahulu

No	Penulis	Judul	Algoritma	Hasil
1	*Chilyabanyama et al.*	*Performance of Machine Learning Classifiers in Classifying Stunting among Under-Five Children in Zambia*	*Logistic Regression, Random Forest, SV classification, XG Boost, Naive Bayes*	*Random Forest* mencapai akurasi tertinggi dengan 79% pada data latih dan 61,6% pada data uji.
2	Hemo, S. A., & Rayhan, M. I. B.	*Classification tree and random forest model to predict under-five malnutrition in Bangladesh*	*Classification Tree, Random Forest*	*Random Forest* mencapai akurasi 70,1% dan AUC 69,8% dalam memprediksi *stunting* pada balita.
3	Talukder, A., & Ahammed, B.	*Machine learning algorithms for predicting malnutrition among under-five children in Bangladesh*	*LDA, KNN, SVM, Random Forest, Logistic Regression*	*Random Forest* memberikan kinerja terbaik dengan akurasi 68,51%, sensitivitas 94,66%, dan spesifisitas 69,76%

Pada penelitian pertama berjudul ”Performance of Machine Learning Classifiers in Classifying Stunting among Under-Five Children in Zambia” oleh *Chilyabanyama et al.*, peneliti membandingkan berbagai algoritma *machine learning* seperti *Logistic Regression, Random Forest, Support Vector Machine, XGBoost,* dan *Naive Bayes* untuk mengklasifikasikan *stunting* pada anak di bawah lima tahun di Zambia [14]. Penelitian ini menggunakan *dataset* kesehatan anak yang dikumpulkan dari survei nasional di Zambia [14]. Hasil penelitian menunjukkan bahwa algoritma *Random Forest* menunjukkan performa terbaik dengan tingkat akurasi sebesar 79% pada data latih dan 61,6% pada data uji [14].

Penelitian kedua berjudul ”Classification Tree and Random Forest Model to Predict Under-Five Malnutrition in Bangladesh” oleh Hemo, S. A. dan Rayhan, M. I. B. memanfaatkan algoritma *Classification Tree* dan *Random Forest* untuk mengklasifikasikan malnutrisi pada balita di Bangladesh [15]. *Dataset* yang digunakan dalam penelitian ini mencakup variabel seperti usia, berat badan, tinggi badan, dan faktor sosial ekonomi [15]. Hasilnya menunjukkan bahwa *Random Forest* mencapai akurasi sebesar 70,1% dan *Area Under the Curve* (AUC) 69,8% dalam memprediksi kasus *stunting* [15].

Pada penelitian ketiga yang berjudul ”Machine Learning Algorithms for Predicting Malnutrition among Under-Five Children in Bangladesh” oleh Talukder, A. dan Ahammed, B., peneliti menggunakan algoritma LDA, KNN, SVM, *Random Forest*, dan *Logistic Regression* untuk memprediksi malnutrisi pada anak-anak di Bangladesh [16]. *Dataset* yang digunakan berasal dari survei *Bangladesh Demographic and Health Survey* yang berisi informasi demografis, kesehatan, serta status gizi anak-anak [16]. Hasil penelitian menunjukkan bahwa algoritma *Random Forest* memberikan kinerja yang terbaik dengan tingkat akurasi sebesar 68,51%, sensitivitas mencapai 94,66%, dan spesifisitas sebesar 69,76% [16].

2.2 Stunting

Stunting merupakan salah satu permasalahan malnutrisi yang serius dan memberikan dampak luas terhadap anak-anak di seluruh dunia [1]. *Stunting* terjadi ketika pertumbuhan anak terhambat akibat kekurangan gizi kronis, yang mengakibatkan tinggi badan anak lebih rendah dibandingkan dengan anak-anak seusianya [3, 17]. Kondisi ini dapat mengganggu fungsi tubuh dan perkembangan kognitif, sehingga memengaruhi kemampuan anak untuk belajar, berpikir logis, dan memiliki kekuatan fisik yang optimal [2].

Penyebab utama *stunting* adalah kekurangan asupan gizi yang berlangsung dalam jangka waktu yang panjang, terutama selama periode seribu hari pertama kehidupan, yang meliputi masa kehamilan hingga usia dua tahun [18]. Selain itu, faktor risiko lainnya termasuk sanitasi yang buruk, infeksi berulang, dan kurangnya akses ke layanan kesehatan yang memadai juga berperan dalam meningkatkan risiko *stunting* [19]. Gejala *stunting* sering kali tidak terlihat jelas pada tahap awal, tetapi dampaknya bisa menjadi nyata saat anak tumbuh. Anak-anak yang mengalami *stunting* cenderung menunjukkan performa akademis yang rendah serta mengalami penurunan produktivitas di masa depan [2, 3].

Pemeriksaan dini dan intervensi yang sesuai sangat krusial dalam upaya penanganan *stunting* [20]. Metode umum yang digunakan untuk menilai *stunting* adalah dengan mengukur tinggi badan anak berdasarkan usia. Pendekatan ini menggunakan *Z-score* tinggi badan terhadap usia (*HAZ*), dimana *HAZ* <-2 menunjukkan *stunting*, dan *HAZ* <-3 menunjukkan *stunting* parah [20, 21]. Penanganan *stunting* melibatkan perbaikan asupan gizi, peningkatan akses ke layanan kesehatan, dan edukasi mengenai pentingnya pola makan seimbang serta kebersihan lingkungan [22].

Meskipun berbagai upaya telah dilaksanakan untuk menurunkan angka *stunting*, kondisi ini tetap menjadi tantangan kesehatan masyarakat yang memerlukan perhatian serius [23]. Oleh karena itu, penelitian senantiasa dilakukan untuk mengembangkan metode deteksi dini serta intervensi yang efektif demi meningkatkan kualitas hidup anak-anak yang berisiko mengalami *stunting* [21, 24].

2.3 Data Splitting

Pembagian data (*Data Splitting*) merupakan metode yang umum diterapkan dalam validasi model, di mana *dataset* dibagi menjadi dua *subset* terpisah: yaitu *subset* pelatihan (*training set*) yang digunakan untuk melatih model, dan *subset* pengujian (*testing set*) yang digunakan untuk mengevaluasi performa model tersebut [25]. Teknik ini bertujuan untuk menghindari *overfitting*, yaitu kondisi dimana model memiliki performa tinggi pada data latih tetapi kurang mampu melakukan generalisasi terhadap data baru [26]. Salah satu penelitian membandingkan berbagai rasio pembagian *dataset*, seperti 80/20, 70/30, 60/40, dan 50/50, dan menemukan bahwa rasio 70/30 dengan *10-fold cross-validation* memberikan performa terbaik [27]. Penelitian lain juga menerapkan rasio 70% untuk data pelatihan dan 30% untuk data pengujian dalam proses seleksi fitur

menggunakan algoritma *Random Forest*. Pendekatan ini diterapkan pada data berdimensi tinggi dengan tujuan untuk meningkatkan performa metode tersebut [28].

2.4 Synthetic Minority Over-sampling Technique

Synthetic Minority Over-sampling Technique (SMOTE) merupakan suatu metode dalam bidang pembelajaran mesin yang diterapkan untuk mengatasi masalah ketidakseimbangan kelas pada sebuah *dataset* [29]. Metode ini bekerja dengan menciptakan sampel sintetis baru dengan menginterpolasi antara *instance* minoritas yang sudah ada [30]. Dengan menambahkan sampel sintetis, SMOTE membantu model pembelajaran mesin untuk memiliki akurasi yang lebih baik dalam mengklasifikasikan data dari kelas minoritas [30, 31].

2.5 Ensemble Learning

Ensemble Learning adalah suatu pendekatan dalam bidang pembelajaran mesin yang mengintegrasikan prediksi dari berbagai model atau algoritma yang berbeda [32, 33]. Dengan mengintegrasikan hasil prediksi dari berbagai model, metode ini bertujuan untuk meningkatkan kestabilan serta akurasi dibandingkan dengan penggunaan model tunggal [34]. Berbagai metode *Ensemble Learning* dapat ditemukan dalam referensi berikut [35, 36, 37, 38].

1. *Bagging*

   Bagging merupakan suatu metode yang menggunakan sejumlah model untuk mencapai keputusan akhir dengan menggabungkan hasil prediksi dari masing-masing model melalui suatu proses *voting*. Pelatihan model dilakukan secara bersamaan dan terpisah, memungkinkan setiap model untuk saling melengkapi kelemahan model lainnya. Salah satu implementasi dari metode *Bagging* adalah algoritma *Random Forest*.

2. *Boosting*

   Boosting merupakan suatu metode yang berorientasi pada peningkatan kinerja dengan memperbaiki kesalahan yang dihasilkan oleh model sebelumnya. Pada metode ini, model dibangun secara berurutan, dimana setiap model berikutnya berusaha untuk mengoreksi kesalahan klasifikasi yang dilakukan oleh model sebelumnya. Salah satu contoh penerapan metode

Boosting adalah *AdaBoost* (*Adaptive Boosting*), yang memberikan bobot lebih besar pada data yang salah diklasifikasikan oleh model sebelumnya.

3. *Stacking*

   Stacking merupakan suatu metode yang mengombinasikan hasil prediksi dari beberapa model untuk membentuk fitur baru yang selanjutnya digunakan dalam melatih model tambahan yang disebut *meta-learner*. Dalam penerapannya, *Stacking* dapat melibatkan berbagai model dasar seperti *Decision Trees*, *Support Vector Machines*, dan *Neural Networks*. Hasil prediksi dari model-model ini kemudian digabungkan dan digunakan sebagai input untuk melatih *meta-learner*, seperti *Logistic Regression*, guna meningkatkan akurasi prediksi.

2.6 Random Forest

Algoritma *Random Forest* merupakan suatu metode dalam pembelajaran mesin yang digunakan untuk mengatasi permasalahan klasifikasi maupun regresi [39, 40]. Algoritma ini membangun sejumlah pohon keputusan dengan memanfaatkan *subset* acak dari *dataset* pelatihan [41]. Secara umum, langkah-langkah berikut digunakan dalam menjalankan algoritma *Random Forest* [42, 43].

1. Pembentukan Pohon Keputusan (*Decision Trees*)

   Dalam algoritma *Random Forest*, setiap pohon keputusan dibangun secara independen satu sama lain. Pohon-pohon ini berfungsi untuk mengelompokkan data dari data pelatihan, sebagaimana yang diperlihatkan pada Gambar 2.1. Proses utama dalam algoritma ini melibatkan pembuatan sejumlah pohon keputusan yang secara kolektif berkontribusi dalam menentukan klasifikasi data secara keseluruhan.

2. Pengambilan Sampel Acak (*Bootstrapping*)

   Sebelum membangun setiap pohon, algoritma *Random Forest* melaksanakan proses *bootstrapping*, yaitu suatu metode pengambilan sampel secara acak dengan penggantian dari data pelatihan. Melalui metode ini, sejumlah data dapat terpilih lebih dari satu kali untuk melatih setiap pohon, yang menghasilkan variasi pada *output* dari masing-masing pohon.

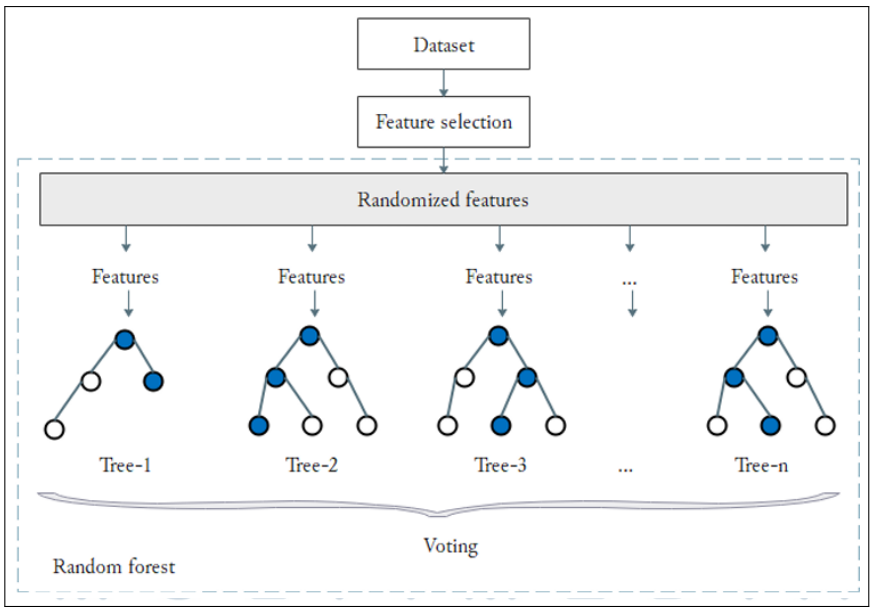
3. Seleksi Fitur Secara Acak (*Random Feature Selection*)

Saat membangun setiap pohon keputusan, hanya sejumlah kecil fitur dari *dataset* yang dipilih secara acak untuk digunakan. Pendekatan ini bertujuan untuk mengurangi korelasi antar pohon serta meminimalkan risiko terjadinya *overfitting*.

4. Pengambilan Keputusan Berdasarkan Suara Terbanyak (*Majority Voting*)

   Setelah seluruh pohon keputusan selesai dibangun, hasil prediksi dari setiap pohon akan digabungkan. Dalam konteks klasifikasi, keputusan akhir ditentukan melalui metode pemungutan suara mayoritas, di mana kelas yang paling banyak dipilih oleh pohon-pohon tersebut menjadi hasil akhir, seperti yang ditunjukkan pada Gambar 2.1. Sementara itu, pada kasus regresi, rata-rata dari semua prediksi pohon digunakan untuk menghasilkan keputusan akhir.

Dengan demikian, algoritma *Random Forest* mampu meminimalisir risiko *overfitting* dan memiliki kemampuan yang baik dalam menangani data berdimensi tinggi, menjadikannya efektif untuk mengolah *dataset* dengan banyak fitur atau atribut [14, 28].



Gambar 2.1. *Feature Selection for Random Forest*
Sumber: [28]

2.7 Confusion Matrix

Confusion Matrix merupakan suatu metode yang digunakan untuk mengevaluasi kinerja model berdasarkan hasil prediksi pada *dataset* uji. [44]. Matriks ini menyajikan hasil klasifikasi model dalam bentuk tabel yang mencocokkan nilai prediksi dengan nilai aktual [45, 46]. Jumlah baris dan kolom pada *Confusion Matrix* bergantung pada jumlah kelas yang ada. Sebagai ilustrasi, apabila terdapat empat kelas, maka *Confusion Matrix* akan terdiri dari empat baris dan empat kolom, sebagaimana ditampilkan pada Tabel 2.2 [47, 48].

Tabel 2.2. Representasi Umum *Confusion Matrix* 4x4

Confusion Matrix		Predicted Value				
		A	B	C	D	
Actual Values	A	TP_A	E_{AB}	E_{AC}	E_{AD}	N_A
	B	E_{BA}	TP_B	E_{BC}	E_{BD}	N_B
	C	E_{CA}	E_{CB}	TP_C	E_{CD}	N_C
	D	E_{DA}	E_{DB}	E_{DC}	TP_D	N_D
		P_A	P_B	P_C	P_D	

Jumlah data yang diuji adalah

$$N_A + N_B + N_C + N_D = P_A + P_B + P_C + P_D = N$$

kelas berdasarkan *Actual Values* adalah

$$N_A = TP_A + E_{AB} + E_{AC} + E_{AD} = TP_A + FN_A$$
$$N_B = TP_B + E_{BA} + E_{BC} + E_{BD} = TP_B + FN_B$$
$$N_C = TP_C + E_{CA} + E_{CB} + E_{CD} = TP_C + FN_C$$
$$N_D = TP_D + E_{DA} + E_{DB} + E_{DC} = TP_D + FN_D$$

Sampel kelas dari *Predicted Values* adalah

$$P_A = TP_A + E_{BA} + E_{CA} + E_{DA} = TP_A + FP_A$$
$$P_B = TP_B + E_{AB} + E_{CB} + E_{DB} = TP_B + FP_B$$
$$P_C = TP_C + E_{AC} + E_{BC} + E_{DC} = TP_C + FP_C$$
$$P_D = TP_D + E_{AD} + E_{BD} + E_{CD} = TP_D + FP_D$$

Metrik pertama yang perlu diperhatikan adalah akurasi keseluruhan dari model klasifikasi [49]. Akurasi dihitung dengan membagi jumlah sampel yang diklasifikasikan dengan benar (TP_A, TP_B, TP_C, TP_D) dengan total jumlah sampel yang diuji (N) [46]. Rumus (2.1) menggambarkan cara perhitungan *Accuracy*.

$$Accuracy = \frac{TP_A + TP_B + TP_C + TP_D}{N} \tag{2.1}$$

Tabel 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, dan 2.10 menyajikan representasi *Confusion Matrix* dalam format 2x2 serta ilustrasi *Confusion Matrix* 4x4 yang digunakan untuk memperoleh kombinasi nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) untuk setiap kelas [46, 50]. Nilai-nilai ini menjadi dasar perhitungan dalam menentukan metrik evaluasi seperti *Accuracy*, *Precision*, *Recall*, dan *F1-Score*, baik untuk masing-masing kelas maupun secara keseluruhan [48, 49, 50].

Tabel 2.3. *Confusion Matrix* 2x2 untuk Kelas A yang Diambil dari *Confusion Matrix* 4x4

Confusion Matrix		Predicted Value	
		A	Bukan A
Actual Values	A	TP_A	$FN_A = E_{AB} + E_{AC} + E_{AD}$
	Bukan A	$FP_A = E_{BA} + E_{CA} + E_{DA}$	$TN_A = TP_B + TP_C + TP_D + E_{BC} + E_{BD} + E_{CD}$

Tabel 2.4. Representasi *Confusion Matrix* 4x4 Kelas A berdasarkan *Confusion Matrix* 4x4

Confusion Matrix		Predicted Value			
		A	B	C	D
Actual Values	A	TP_A	FN_A	FN_A	FN_A
	B	FP_A	TN_A	TN_A	TN_A
	C	FP_A	TN_A	TN_A	TN_A
	D	FP_A	TN_A	TN_A	TN_A

Tabel 2.5. *Confusion Matrix* 2x2 Kelas B yang Diambil dari *Confusion Matrix* 4x4

Confusion Matrix		Predicted Value	
		B	Bukan B
Actual Values	B	TP_B	$FN_B = E_{BA} + E_{BC} + E_{BD}$
	Bukan B	$FP_B = E_{AB} + E_{CB} + E_{DB}$	$TN_B = TP_A + TP_C + TP_D + E_{AC} + E_{AD} + E_{CD}$

Tabel 2.6. Representasi *Confusion Matrix* 4x4 Kelas B berdasarkan *Confusion Matrix* 4x4

Confusion Matrix		Predicted Value			
		A	B	C	D
Actual Values	A	TN_B	FP_B	TN_B	TN_B
	B	FN_B	TP_B	FN_B	FN_B
	C	TN_B	FP_B	TN_B	TN_B
	D	TN_B	FP_B	TN_B	TN_B

Tabel 2.7. *Confusion Matrix* 2x2 Kelas C yang Diambil dari *Confusion Matrix* 4x4

Confusion Matrix		Predicted Value	
		C	Bukan C
Actual Values	C	TP_C	$FN_C = E_{CA} + E_{CB} + E_{CD}$
	Bukan C	$FP_C = E_{AC} + E_{BC} + E_{DC}$	$TN_C = TP_A + TP_B + TP_D + E_{AB} + E_{AD} + E_{BD}$

Tabel 2.8. Representasi *Confusion Matrix* 4x4 Kelas C berdasarkan *Confusion Matrix* 4x4

Confusion Matrix		Predicted Value			
		A	B	C	D
Actual Values	A	TN_C	TN_C	FP_C	TN_C
	B	TN_C	TN_C	FP_C	TN_C
	C	FN_C	FN_C	TN_C	FN_C
	D	TN_C	TN_C	FP_C	TN_C

Tabel 2.9. *Confusion Matrix* 2x2 Kelas D yang Diambil dari *Confusion Matrix* 4x4

Confusion Matrix		Predicted Value	
		D	Bukan D
Actual Values	D	TP_D	$FN_D = E_{DA} + E_{DB} + E_{DC}$
	Bukan D	$FP_D = E_{AD} + E_{BD} + E_{CD}$	$TN_D = TP_A + TP_B + TP_C + E_{AB} + E_{AC} + E_{BC}$

Tabel 2.10. Representasi *Confusion Matrix* 4x4 Kelas D berdasarkan *Confusion Matrix* 4x4

Confusion Matrix		Predicted Value			
		A	B	C	D
Actual Values	A	TN_D	TN_D	TN_D	FP_D
	B	FN_D	TN_D	TN_D	FP_D
	C	TN_D	TN_D	TN_D	FP_D
	D	FN_D	FN_D	FN_D	TP_D

Berikut adalah metode perhitungan *Confusion Matrix* yang digunakan untuk menentukan nilai *Precision*, *Recall*, dan *F1-Score* pada setiap kelas [46].

1. *Precision*

   Precision digunakan untuk mengukur sejauh mana prediksi positif yang dihasilkan oleh model sesuai dengan kondisi sebenarnya, atau seberapa banyak prediksi positif yang benar. Dengan kata lain, *Precision* mengevaluasi tingkat akurasi model dalam mengidentifikasi kelas positif. Rumus (2.2), (2.3), (2.4), dan (2.5) menunjukkan metode perhitungan *Precision* untuk setiap kelas, sedangkan Rumus (2.6) digunakan untuk menghitung *Precision* secara keseluruhan.

$$Precision_A = \frac{TP_A}{TP_A + FP_A} = \frac{TP_A}{P_A} \tag{2.2}$$

$$Precision_B = \frac{TP_B}{TP_B + FP_B} = \frac{TP_B}{P_B} \tag{2.3}$$

$$Precision_C = \frac{TP_C}{TP_C + FP_C} = \frac{TP_C}{P_C} \tag{2.4}$$

$$Precision_D = \frac{TP_D}{TP_D + FP_D} = \frac{TP_D}{P_D} \tag{2.5}$$

$$Precision = \frac{Precision_A + Precision_B + Precision_C + Precision_D}{4} \tag{2.6}$$

2. *Recall*

   Recall mengukur kemampuan model dalam mengidentifikasi semua nilai positif yang sebenarnya. Metrik ini mengevaluasi seberapa baik model dalam menemukan seluruh *instance* yang termasuk dalam kelas positif. Rumus (2.7), (2.8), (2.9), dan (2.10) menjelaskan cara menghitung *Recall* untuk setiap kelas, sedangkan Rumus (2.11) digunakan untuk memperoleh nilai *Recall* secara keseluruhan.

$$Recall_A = \frac{TP_A}{TP_A + FN_A} = \frac{TP_A}{N_A} \tag{2.7}$$

$$Recall_B = \frac{TP_B}{TP_B + FN_B} = \frac{TP_B}{N_B} \tag{2.8}$$

$$Recall_C = \frac{TP_C}{TP_C + FN_C} = \frac{TP_C}{N_C} \tag{2.9}$$

$$Recall_D = \frac{TP_D}{TP_D + FN_D} = \frac{TP_D}{N_D} \tag{2.10}$$

$$Recall = \frac{Recall_A + Recall_B + Recall_C + Recall_D}{4} \tag{2.11}$$

3. *F1-Score*

   F1-Score adalah metrik evaluasi yang menggabungkan *Precision* dan *Recall* untuk memberikan penilaian yang menyeluruh terhadap kinerja model. Nilai *F1-Score* yang tinggi mengindikasikan bahwa model memiliki ketepatan dan kemampuan deteksi positif yang baik. Rumus (2.12), (2.13), (2.14), dan (2.15) merepresentasikan perhitungan *F1-Score* untuk setiap kelas, sedangkan Rumus (2.16) digunakan untuk menghitung *F1-Score* secara keseluruhan.

$$F1\text{-}Score_A = \frac{2 \cdot Precision_A \cdot Recall_A}{Precision_A + Recall_A} \tag{2.12}$$

$$F1\text{-}Score_B = \frac{2 \cdot Precision_B \cdot Recall_B}{Precision_B + Recall_B} \tag{2.13}$$

$$F1\text{-}Score_C = \frac{2 \cdot Precision_C \cdot Recall_C}{Precision_C + Recall_C} \tag{2.14}$$

$$F1\text{-}Score_D = \frac{2 \cdot Precision_D \cdot Recall_D}{Precision_D + Recall_D} \tag{2.15}$$

$$F1\text{-}Score = \frac{F1\text{-}Score_A + F1\text{-}Score_B + F1\text{-}Score_C + F1\text{-}Score_D}{4} \tag{2.16}$$