

# **Analisis Penilaian Esai Otomatis dengan Menggunakan Model *Multivariate Regression* dan Metode *Wrapper* untuk *Features Selection***



**Institut Teknologi Del**

**2016/2017**

# **Lembar Pengesahan Tugas Akhir Institut Teknologi Del**

**Analisis Penilaian Esai Otomatis dengan Menggunakan Model  
*Multivariate Regression* dan Metode *Wrapper* untuk *Features Selection***

**Oleh :**

**institut teknologi**

11314058  
11314060  
11314062

Aditya Pranata Siregar  
May Kana Sagala  
Taufan Silitonga

Sitoluama, September 2017

Pembimbing

Roy Deddy Hasiholan Lumban Tobing, S.T., M.ICT.

NIDN:0121038401

**MARTUHAN-MARROHA-MARBISUK**

**2001**

**Dinyatakan memenuhi syarat dan karenanya disetujui dan disahkan sebagai**

**Laporan Tugas Akhir Diploma 3**

**Program Studi Teknik Informatika**

**Institut Teknologi Del**

## Prakata

Puji dan syukur kami panjatkan kepada Tuhan Yang Maha Esa atas rahmat-Nya menyertai penulis selama proses penulisan laporan Tugas Akhir ini. Sehingga setiap tahapan yang direncanakan untuk penyusunan laporan Tugas Akhir ini dapat dilalui dengan baik.

Tujuan penelitian pada laporan ini adalah untuk melakukan analisis terhadap penilaian esai otomatis dengan menggunakan model *multivariate regression* dan metode *wrapper* untuk *features selection*. Penulis mengucapkan terimakasih kepada Bapak Roy Deddy Hasiholan Lumban Tobing sebagai dosen pembimbing, yang telah memberikan arahan dan bimbingan dalam menyelesaikan Tugas Akhir ini. Ucapan terimakasih juga penulis sampaikan kepada Koordinator Tugas Akhir 2016/2017, Bapak penguji dan semua pihak yang tidak dapat disebutkan satu per satu, atas bantuan yang diberikan selama penyusunan laporan Tugas Akhir ini.

Penulis mengharapkan laporan Tugas Akhir ini dapat memberikan informasi mengenai penilaian esai otomatis dengan menggunakan model *multivariate regression* dan metode *wrapper* untuk *features selection*. Penulis juga menyadari bahwa laporan ini tidak lepas dari kesalahan , oleh karena itu penulis mengharapkan kritik dan saran yang membangun untuk perbaikan laporan Tugas Akhir ini.

Sitoluama, September 2017

2001

11314058 Aditya Pranata Siregar

11314060 May Kana Sagala

11314062 Taufan Silitonga

## Abstrak

*Automated Essay Grading* merupakan proses penilaian esai yang dilakukan dengan bantuan program komputer sehingga proses tersebut menjadi otomatis dan membutuhkan biaya dan waktu lebih sedikit. *Automated Essay Grading* dapat dilakukan dengan pengimplementasian model prediktif dengan beberapa algoritma *Machine Learning*. Berdasarkan pertimbangan-pertimbangan tertentu, penulis memilih mengimplementasikan *multivariate linear regression* dan *multivariate logistic regression* dengan *feature* seperti *total word count* per esai, *sentence count*, *number of long words*, *part of speech counts*, dsb. Penulis juga mengimplementasikan *forward* dan *wrapper features selection* untuk pemilihan *feature* dengan kontribusi paling banyak terhadap akurasi model prediktif, serta menambahkan parameter *n-grams* dan menganalisa kontribusi parameter tersebut terhadap akurasi model prediktif yang diukur dengan menggunakan *root mean squared error* dan *quadratic weighted kappa*.

Berdasarkan eksperimen yang dilakukan, penulis menyimpulkan bahwa *multivariate linear regression* bekerja lebih baik daripada *multivariate logistic regression* dan penulis pun menyimpulkan bahwa *wrapper method* bekerja lebih baik daripada *forward features selection*, serta penambahan parameter *n-grams* dapat meningkatkan akurasi model prediktif.

**Kata Kunci :** *Automated Essay Grading*, *Multivariate Linear Regression*, *Multivariate Logistic Regression*, *Forward and Wrapper Feature Selection*, *RMSE*, *Quadratic Weighted Kappa*.

# DAFTAR ISI

<b>Prakata .....</b>	.3
<b>Abstrak.....</b>	.4
<b>Bab I Pendahuluan .....</b>	.9
1.1 Latar Belakang .....	.9
1.2 Tujuan .....	.11
1.3 Lingkup .....	.12
1.4 Pendekatan .....	.13
1.5 Sistematika Penyajian .....	.15
<b>Bab II Tinjauan Pustaka .....</b>	.16
2.1 Esai.....	.16
2.2 Penilaian Esai .....	.16
2.2.1 Penilaian Esai Secara manual .....	.16
2.2.2 Penilaian Esai Secara otomatis .....	.16
2.3 Machine Learning .....	.17
2.4 Text Mining.....	.19
2.4.1 N-Grams .....	.19
2.5 Linear Regression.....	.20
2.6 Logistic Regression Model.....	.21
2.7 Multivariate Linear Regression .....	.21
2.8 Root Mean Square Error (RMSE).....	.21
2.9 Quadratic Weighted Kappa .....	.22
2.10 Overfitting dan Underfitting.....	.22
2.11 Cross Validation .....	.23
2.12 Preprocessing.....	.23
2.13 Features .....	.24
2.14 Feature Extraction .....	.24
2.15 Features Selection.....	.25
2.15.1 Kegunaan Features Selection.....	.25
2.15.2 Algoritma Features Selection.....	.26
2.16 Features Scaling .....	.26
2.17 Current Research .....	.27
2.18 Ringkasan .....	.27
<b>Bab III Analisis .....</b>	.28
3.1 Analisis Data .....	.28
3.2 Prosedur dan Standar Penilaian Manual.....	.31
3.2.1 Prompt(Kata Pengantar) .....	.31
3.2.2 Instruction(Instruksi) .....	.31
3.2.3 Rubic Guidelines(Standar Penilaian) .....	.31
3.2.4 Adjudication Rules(Penentuan Skor Final) .....	.33
3.3 Analisis Preprocessing.....	.33
3.3.1 Case Folding.....	.33
3.3.2 Tokenization .....	.34
3.3.3 Stopword Removal .....	.34
3.4 Analisis Feature Extraction .....	.35
3.5 Analisis Features Selection .....	.36
3.6 Analisis Feature Scaling.....	.38
3.7 Model Evaluation.....	.38
3.7.1 RMSE .....	.38
3.7.2 Quadratic Weighted Kappa.....	.40
3.8 Analisis Training Model.....	.41
3.8.1 Regresi Linear.....	.41
3.8.2 Logistic Regression Model .....	.44
3.9 Ringkasan.....	.47
<b>Bab IV Desain dan Implementasi Eksperimen .....</b>	.48
4.1 Deskripsi Umum .....	.48
4.2 Lingkungan Eksperimen .....	.48

4.3	Tahapan Eksperimen .....	49
4.3.1	Tahapan Eksperimen terhadap metode Wrapper pada proses features selection.....	49
4.3.2	Tahapan Eksperimen terhadap penambahan parameter n-grams.....	52
4.4	Deskripsi Program.....	53
4.4.1	Prepocessing .....	53
4.4.2	Feature Extraction.....	54
4.4.3	Features Selection.....	57
4.4.4	Training Model .....	58
<b>Bab V</b>	<b>Hasil dan Pembahasan .....</b>	<b>59</b>
5.1	Distribusi Data .....	59
5.1.1	Word and Sentence .....	59
5.1.2	Part of Speech Tagger.....	60
5.1.3	Structure and Orthography .....	61
5.1.4	Foreign Word and Lexical Diversity .....	62
5.1.5	Misspelt Error .....	64
5.2	Single Feature Model Evaluation .....	64
5.2.1	Linier Regression Model .....	65
5.2.2	Logistic Regression Model .....	66
5.3	Forward Feature Selection .....	66
5.3.1	Linear Regression Model Measured by Kappa Score.....	66
5.3.2	Linear Regression Model Measured by RMSE Score .....	68
5.3.3	Logistic Regression Model Measured by Kappa Score .....	70
5.3.4	Logistic Regression Model Measured by RMSE Score.....	71
5.4	Penambahan Parameter N-grams terhadap Forward Method.....	73
5.5	Wrapper Features Selection .....	73
5.6	Penambahan Parameter N-grams terhadap Wrapper Method .....	74
<b>Bab VI</b>	<b>Kesimpulan dan Saran.....</b>	<b>76</b>
6.1	Kesimpulan .....	76
6.2	Saran .....	76
<b>Daftar Pustaka .....</b>		<b>77</b>
<b>Lampiran.....</b>		<b>79</b>

## Daftar Gambar

Gambar 1. Metodologi Penelitian.....	13
Gambar 2. Proses <i>text mining</i> .....	19
Gambar 3. <i>Overfitting</i> dan <i>underfitting</i> .....	22
Gambar 4. Proses <i>case folding</i> .....	33
Gambar 5. Proses <i>tokenizing</i> dengan tanda spasi sebagai <i>token</i> pemisah.....	34
Gambar 6. Tingkat akurasi .....	36
Gambar 7. Proses <i>training model</i> .....	41
Gambar 8. Tahapan eksperimen terhadap metode <i>wrapper</i> pada proses <i>features selection</i> .....	51
Gambar 9. Tahapan eksperimen terhadap penambahan parameter <i>n-grams</i> .....	53
Gambar 10. <i>Numerical features</i> .....	59
Gambar 11. <i>Part of speech tagger</i> .....	60
Gambar 12. <i>Structure and orthography</i> .....	61
Gambar 13. <i>Overall punctuation count</i> .....	62
Gambar 14. <i>Foreign word count</i> .....	62
Gambar 15. <i>Lexical diversity count</i> .....	63
Gambar 16. <i>Orthography misspelt errors</i> .....	64
Gambar 17. <i>Linear regression model</i> .....	65
Gambar 18. <i>Logistic regression model</i> .....	66
Gambar 19. Pengurutan dari nilai tingkat akurasi yang tertinggi sampai ke terendah.....	67
Gambar 20. Proses <i>forward selection</i> .....	67
Gambar 21. Pengurutan dari nilai RMSE yang rendah sampai ke tinggi .....	68
Gambar 22. Proses <i>forward feature RMSE</i> .....	69
Gambar 23. Perhitungan pada kappa <i>score</i> regresi <i>logistic</i> .....	70
Gambar 24. Hasil <i>forward selection</i> .....	71
Gambar 25. Perhitungan pada RMSE <i>score</i> regresi <i>logistic</i> .....	72
Gambar 26. Hasil dari pada <i>forward selection</i> .....	72
Gambar 27. Hasil penambahan parameter <i>n-grams</i> terhadap <i>forward method</i> .....	73
Gambar 28. Proses <i>wrapper features selection</i> .....	74
Gambar 29. Penambahan parameter <i>n-grams</i> terhadap <i>wrapper method</i> .....	75

## Daftar Tabel

Tabel 1. <i>N-grams</i> berbasis karakter.....	.19
Tabel 2. <i>N-grams</i> berbasis kata .....	.20
Tabel 3. Hasil prediksi.....	.39
Tabel 4. Hitung nilai e .....	.39



## Bab I Pendahuluan

Bab ini berisi uraian mengenai latar belakang pemilihan topik, tujuan, lingkup pelaksanaan Tugas Akhir, dan sistematika penyajian dalam menyelesaikan persoalan yang menjadi objek Tugas Akhir.

### 1.1 Latar Belakang

Esai dapat melatih seseorang dalam penyampaian suatu informasi secara verbal untuk mendapatkan pemahaman yang lebih mendalam. Penilaian memiliki peran yang sangat penting dalam dunia pendidikan. Pada kasus ini penilaian akan dilakukan terhadap esai yang terdiri atas beberapa paragraf. Penulisan esai tidak luput dari kesalahan yang tidak disengaja baik dalam penulisan, tanda baca, dan pemilihan kata. Proses penilaian terhadap esai oleh tim penilai sangat dibutuhkan untuk memperoleh esai yang lebih baik. Dalam proses penilaian, tim penilai akan sangat kesulitan dalam melakukan proses penilaian terhadap esai dikarenakan banyaknya jumlah esai yang akan diperiksa. Pada kasus ini komputer akan berperan sangat penting dalam melakukan penilaian terhadap esai.

Fungsi dasar komputer seperti *word processor* telah memberikan bantuan besar kepada penulis dalam proses modifikasi esai serta proses pendokumentasian tulisan-tulisan mereka. Pada dasarnya, proses penilaian esai yang dilakukan secara manual membutuhkan proses yang sangat lama, tidak konsisten, dan membutuhkan banyak sumber daya. Oleh sebab itu, dengan bantuan program tertentu penulis dapat dengan mudah melakukan proses pemeriksaan ejaan, dan penggunaan tanda baca. Sedangkan untuk penilaian konten esai dibutuhkan teknik khusus karena parameter penilaian terhadap esai bersifat dinamis.

Teknik khusus yang digunakan adalah *Automated Essay Scoring*(AES). Pada AES telah dijelaskan bahwa komputer memiliki fungsi sebagai alat kognitif yang lebih efektif (DIKLI, 2006) dalam proses penilaian esai. AES didefinisikan sebagai teknologi komputer yang mengevaluasi dan menilai esai (Shermis & Burstein, 2003) (Shermis & Barrera, 2002) (Shermis & Raymat, 2003). AES telah memiliki banyak pengaplikasian dengan metode yang berbeda-beda. Setiap metode memiliki kelebihan dalam penanganan parameter tertentu akan tetapi, memiliki kekurangan dalam penanganan parameter

lainnya. Hal ini dikarenakan parameter penilaian setiap orang/lembaga tertentu bisa berbeda-beda meskipun sudah terdapat beberapa standarisasi dalam penilaian esai.

Revisi dan umpan balik adalah aspek penting dalam proses penulisan. Penulis perlu menerima umpan balik untuk meningkatkan kualitas tulisannya. Namun, ada beberapa kesulitan yang mungkin muncul yaitu ketika jumlah esai yang akan dinilai cukup banyak. Dalam situasi yang seperti ini, keakuratan, kecepatan dan konsistensi penilaian menjadi sangat penting. Untuk menangani situasi tersebut dibutuhkan teknik khusus untuk mempertahankan ataupun meningkatkan keakuratan, kecepatan dan konsistensi dalam penilaian terhadap esai.

Penilaian esai secara otomatis telah banyak dilakukan seperti yang terdapat pada jurnal *An Overview of Current Research on Automated Essay Grading* (Valenti, et al., 2003) dengan beberapa sistem yang dibangun adalah sebagai berikut : *Project Essay Grader*(PEG), *Intelligent Essay Assessor*(IEA), *Educational Testing Servive*(ETS), *Electronic Essay Rater*(E-Rater), *Conceptual Rater*(C-Rater), *Bayesian Essay Test Scoring System*(BETSY), *Intelligent Essay Marking Systems*(IEMS), *Automark*, *Schema Extract Analyse and Report*(SEAR) dan *Paperless School free-text Marking Engine*(PS-ME) (Valenti, et al., 2003).

*Project Essay Grade*(PEG) merupakan perangkat lunak pertama yang digunakan dalam proses *Automated Essay Grading* (Valenti, et al., 2003). PEG menggunakan metode statistika, dimana esai tidak dinilai berdasarkan kontennya namun dinilai berdasarkan kualitas menulisnya.

*Intelligent Essay Assessor*(IEA) (Valenti, et al., 2003) dikembangkan pada tahun 90-an (Callear, et al., 2001). Sistem penilaianannya menggunakan LSA(*Latent Semantic Analysis*), yang pada dasarnya di desain untuk *indexing* dokumen dan pengambilan teks (Launder, et al., 1990) dengan menggunakan matriks algebra atau disebut dengan istilah *Singular Value Decomposition*(SDV).

*Electronic Essay Rater* (E-Rater) menggunakan metode *lexicon-semantic* dengan *domain-specific, concept-based lexicon* dan konsep penggunaan tata bahasa, akan tetapi hanya bekerja pada fragmen kalimat dengan panjang 15-20 kata (Valenti, et al., 2003).

*Bayesian Essay Test Scoring*(BETSY) menggunakan metode *text clasification*(NativeBayes) (Valenti, et al., 2003).

*Automark* merupakan sebuah perangkat lunak yang kuat terkomputerisasi dengan menandai *free-text answer* untuk pertanyaan-pertanyaan terbuka (Mitchell, et al., 2002). *Automark* menggabungkan sejumlah model pengolahan khusus bertujuan untuk memberikan tanda yang kuat dalam setiap kesalahan, seperti kesalahan ejaan, pengetikan, sintaksis dan semantik.

Selain metode-metode yang disebutkan diatas, *multivariate regression* juga dapat digunakan sebagai *learning model* yang sesuai untuk melakukan prediksi terhadap data non tunggal kontinu dengan berbagai parameter penilaian (*features*) terhadap sebuah esai (Alexopoulos, t.thn.). Pada kenyataannya, tidak semua *features* tersebut digunakan sebagai variabel yang mempengaruhi nilai esai. Maka, perlu dilakukan *features selection* sehingga akan dihasilkan tingkat akurasi prediksi yang paling baik beserta parameter-parameter untuk mendapatkan tingkat akurasi terbaik.

Pada *Automated Essay Grading Using Machine Learning*(Mahana, et al., 2012) dijelaskan bahwa *feature NLP* (*Natural Language Processing*) seperti *n-grams* dan *k-nearest* dipercaya dapat meningkatkan tingkat akurasi prediksi model. Hal ini menjadi salah satu pertimbangan bagi penulis untuk melakukan validasi terhadap *future work* dari penelitian tersebut. Penulis juga menemukan bagian yang dapat dioptimasi dari penelitian tersebut yang kemudian akan dijelaskan pada sub bab 3.5.

## 1.2 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut :

- Melakukan pembuktian dengan dilakukannya *wrapper method* pada saat proses *features selection*, maka akan dicapai tingkat akurasi prediksi skor esai yang lebih baik.
- Melakukan validasi terhadap hasil dari penelitian sebelumnya yang menyatakan bahwa Model *Multivariate Linear Regression* bekerja baik untuk *Automated Essay Grading*(Mahana, et al., 2012).
- Melakukan pembuktian asumsi pada penelitian sebelumnya bahwa dengan penambahan *feature n-grams* maka akan dicapai tingkat akurasi model prediktif yang lebih baik(Mahana, et al., 2012).

Secara keseluruhan untuk mendukung tercapainya tujuan-tujuan diatas, maka digunakan model prediktif *Multivariate Regression* baik tipe *logistic* maupun *linear*. Dan dengan adanya pembuktian hipotesa-hipotesa tersebut, penulis mengharapkan dicapainya sebuah metode penilaian esai otomatis dengan tingkat akurasi yang lebih baik, terkhusus pada esai dengan jenis paragraf narasi, persuasi, dan ekspositori.

### 1.3 Lingkup

Ruang lingkup dari Tugas Akhir ini adalah sebagai berikut :

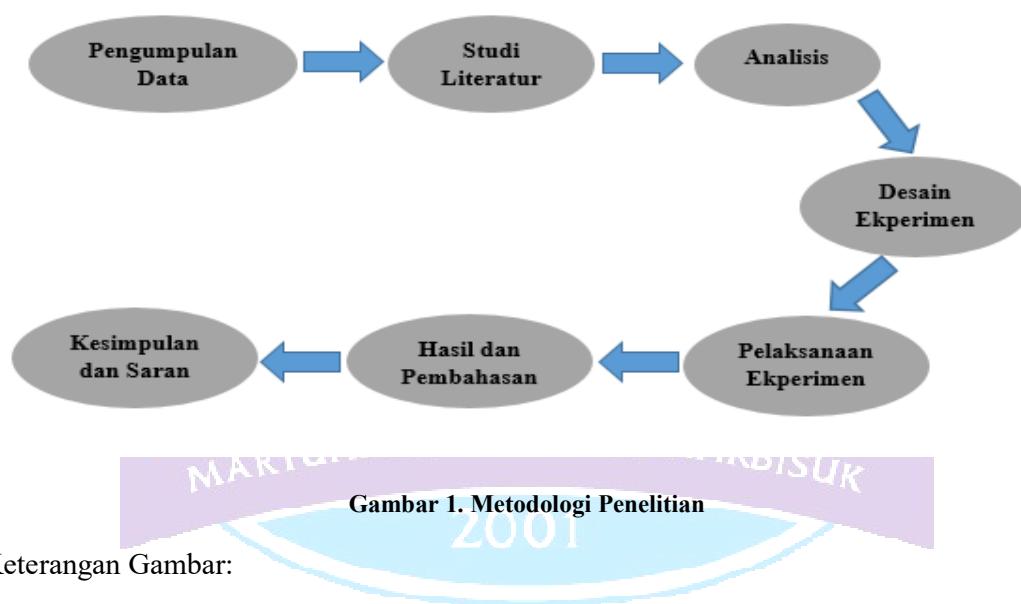
1. Dataset bersumber dari Kaggle.com, oleh *William and Flora Hewlett Foundation* Adapun yang menjadi landasan pertimbangan pengambilan dataset tersebut adalah sebagai berikut :
  - Dataset terdiri dari 1,785 esai dengan tipe naratif, persuasif, dan ekspositori yang sudah memiliki *correct output* (*score\_1*, *score\_2*, *final score*), sehingga memungkinkan dilakukannya *supervised learning* tipe regresi.
  - Esai dari Kaggle.com sudah mengandung *name entity tags* dari *Stanford Name Entity Recognizer Group*, sehingga *name entity* seperti *person*, *organization*, dan *location* sudah dikategorikan sebagai *proper noun*. Hal ini sangat menguntungkan, karena dapat membantu tahap *pre-processing* data.
2. Esai yang digunakan merupakan esai jenis naratif, persuasif, dan ekspositori dalam format ASCII dan ditulis oleh pelajar kelas 7 sampai kelas 10 (Mahana, et al., 2012). Jenis esai ini dipilih karena merupakan jenis esai yang terdapat pada dataset adalah jenis narasi, persuasi, dan ekspositori.
3. Model yang digunakan adalah *multivariate regression* tipe *linear* dan *logistic*, model ini dipilih berdasarkan landasan bahwa model tipe *linear* digunakan pada penelitian sebelumnya dan penulis menambahkan tipe *logistic* sebagai model untuk analisis lebih mendalam.
4. Metode *features selection* yang digunakan adalah *wrapper method*. Metode ini dipilih dengan landasan bahwa dengan menggunakan metode ini, dapat dianalisa seluruh kemungkinan sub-himpunan dari himpunan *features*, sehingga didapatkan sub-himpunan dengan hasil prediksi paling optimal. Akan tetapi dikarenakan keterbatasan

*source code* dari penelitian sebelumnya, penulis tetap mengimplementasikan *forward features selection*.

5. Setiap *source-code* pada penelitian ini berdiri sendiri (*stand-alone*) dan tidak saling terintegrasi menjadi sebuah perangkat lunak *Automated Essay Grading*, dikarenakan tujuan dari penelitian ini bukanlah untuk pembangunan *software*, melainkan untuk melakukan analisis dan validasi terhadap penelitian sebelumnya.

#### 1.4 Pendekatan

Dalam pengerjaan Tugas Akhir ini, penulis melakukan analisis terhadap penilaian esai otomatis dengan menggunakan model *multivariate regression* (*linear* dan *logistic*) dan metode *wrapper* untuk *features selection* dan pengaruh penambahan *n-grams feature* terhadap akurasi prediksi model. Sehingga metodologi yang digunakan dapat dilihat pada gambar 1 dibawah ini :



##### 1. Pengumpulan Data

Tahap pengumpulan data dilakukan untuk memeriksa jumlah ketersediaan data yang akan digunakan dalam proses penelitian. Dataset yang digunakan dalam Tugas Akhir ini diperoleh dari Kaggle.com yang merupakan *Online Judge* bagi para penggiat di bidang *Data Scientist*.

## 2. Studi Literatur

Studi literatur adalah tahap dalam metodologi penelitian yang digunakan untuk mengumpulkan informasi ataupun artikel yang berhubungan dengan topik Tugas Akhir. Tujuan dari studi literatur ini adalah untuk lebih memperdalam pemahaman terhadap *multivariate regression*, *metode wrapper* untuk *features selection* dan *n-grams*. Informasi diperoleh dari artikel, jurnal, buku, *e-course* dan internet.

## 3. Analisis

Tahap ini akan dilakukan analisis terhadap *multivariate regression* (*linear* dan *logistic*) dan metode *wrapper* untuk *features selection*, serta analisis penambahan *n-grams feature*. Adapun analisis lain yang dilakukan adalah penggalian pemahaman yang lebih mendalam terhadap tahapan-tahapan dalam proses pembentukan model, performansi dan akurasi kedua model, analisis data, analisis *model validation*, prosedur dan standar penilaian esai, *text-preprocessing*, dan *feature extraction*.

## 4. Desain Eksperimen

Tahap desain eksperimen ini dilakukan untuk merancang sistematika eksperimen berdasarkan hasil studi literatur dan kebutuhan serta lingkungan eksperimen.

## 5. Hasil dan Pembahasan

Tahap ini akan dijelaskan pembahasan tentang analisis penilaian esai otomatis dengan menggunakan model *multivariate regression* dan metode *wrapper* untuk *features selection*, serta pengaruh penambahan *n-grams feature* terhadap akurasi prediksi model.

## 6. Kesimpulan dan Saran

Tahapan ini akan membahas rangkuman yang diperoleh hasil dari pengumpulan data, studi literatur, analisis, desain eksperimen dan hasil dan pembahasan, serta saran untuk penelitian lebih lanjut.

## 1.5 Sistematika Penyajian

Laporan Tugas Akhir ini terdiri dari enam bab, yaitu:

Pada Bab I berisi pendahuluan yang menguraikan penjelasan mengenai latar belakang, tujuan, lingkup, pendekatan dan sistematika penyajian dari analisis penilaian esai otomatis dengan menggunakan model *multivariate regression* dan metode *wrapper* untuk *features selection*.

Pada Bab II berisi mengenai tinjauan pustaka yang menguraikan dasar-dasar teori mengenai analisis penilaian esai otomatis dengan menggunakan model *multivariate regression* dan metode *wrapper* untuk *features selection*. Dasar teori tersebut meliputi dasar teori esai, penilaian esai, *machine learning*, *text mining*, regresi *linear*, *logistic regression* model, *multivariate linear regression*, *quadratic weighted kappa*, *overfitting* dan *underfitting*, *cross validation*, *preprocessing*, *feature extraction*, *features selection*, *features scaling*, dan ringkasan.

Pada Bab III berisi analisis dari penilaian esai otomatis dengan menggunakan model *multivariate regression* dan metode *wrapper* untuk *features selection*. Analisis tersebut meliputi analisis data, prosedur dan standar penilaian manual, analisis *preprocessing*, analisis *features extraction*, analisis *features selection*, analisis *features scaling*, model *evaluation*, analisis *training model*, dan ringkasan.

Pada Bab IV berisi desain dan implementasi eksperimen dari penilaian esai otomatis dengan menggunakan model *multivariate regression* dan metode *wrapper* untuk *features selection*. Desain eksperimen tersebut meliputi deskripsi umum, lingkungan eksperimen, tahapan eksperimen, dan deskripsi program pada setiap tahapan eksperimen.

Pada Bab V berisi hasil dan pembahasan dari eksperimen yang dilakukan terhadap analisis penilaian esai otomatis. Hasil dan pembahasan tersebut meliputi distribusi data, *single feature selection*, *forward feature selection*, penambahan parameter *n-grams*, dan *wrapper features selection*.

Pada Bab VI berisi kesimpulan dan saran yang disimpulkan berdasarkan data-data yang didapat dari hasil eksperimen yang dilakukan pada tahapan sebelumnya.

## Bab II Tinjauan Pustaka

### 2.1 Esai

Esai adalah karangan prosa yang membahas suatu masalah secara sepintas dari sudut pandang pribadi penulisnya(diambil dari KBBI, 2005) Esai banyak digunakan dalam kegiatan akademik, seperti penerimaan beasiswa dan analisa pemahaman verbal terhadap topik tertentu.

### 2.2 Penilaian Esai

Pada sub bab ini akan dijelaskan penilaian esai yang terbagi atas penilaian esai secara manual dan otomatis.

#### 2.2.1 Penilaian Esai Secara manual

Pada sub bab penilaian esai secara manual ini tim penilai melakukan penilaian terhadap esai secara manual. Pada penilaian esai secara manual, tim penilai melakukan seleksi dengan membaca esai terlebih dahulu dan memberikan penilaian berdasarkan ilmu dan pengetahuan yang telah dimiliki dalam proses penilaian terhadap esai. Kemudian, keseluruhan penilai akan merangkum hasil penilaian masing-masing menjadi sebuah nilai *final*.

#### 2.2.2 Penilaian Esai Secara otomatis

Penilaian esai secara otomatis (AES) merupakan "kemampuan teknologi komputer untuk mengevaluasi dan memberi nilai terhadap karya tulisan berupa prosa berdasarkan pertimbangan-pertimbangan tertentu" (Shermis & Raymat, 2003).

AES diperoleh dengan memberikan pengetahuan kepada teknologi komputer tentang standar ataupun prosedur penilaian yang dapat dilakukan dengan *machine learning*.

AES sendiri memiliki peran sangat penting dalam penilaian esai *modern*, terutama untuk kondisi dimana jumlah esai yang banyak dan lingkungan penilai yang majemuk dikarenakan AES dapat melakukan optimasi biaya/*cost* yang dibutuhkan dalam proses penilaian esai, baik biaya materil maupun non-materil.

## 2.3 Machine Learning

*Machine learning* adalah bagian dari *Artificial Intelligence* (kecerdasan buatan). *Artificial Intelligence* (Kecerdasan Buatan) dapat diterapkan pada sistem komputer atau mesin yang menggunakan sebuah teknik tertentu dalam meniru fungsi kognitif aktivitas otak manusia, seperti belajar dan memecahkan masalah. *Machine learning* merupakan salah satu disiplin ilmu dari *computer science* yang mempelajari bagaimana membuat komputer/mesin itu mempunyai suatu kecerdasan ataupun kemampuan untuk melakukan pembelajaran tanpa harus harus diberi penjelasan secara eksplisit (Chai & Draxler, 2012).

Dengan kata lain *machine learning* merupakan ilmu yang mempelajari bagaimana memberikan kemampuan terhadap komputer untuk melakukan aktifitas belajar dalam menyelesaikan masalah secara mandiri.

Dalam *machine learning* ada beberapa kategori pembelajaran (Chai & Draxler, 2012), yaitu :

### 1. Pembelajaran Terarah (*Supervised Learning*)

*Supervised learning* dapat dikatakan metode pembelajaran dengan adanya latihan. Data latihan ini terdiri dari pasangan nilai *input* dan *output*. *Supervised leaning* ini digunakan untuk memprediksi nilai fungsi untuk nilai semua *input* yang tersedia.

Yang termasuk metode *Supervised leaning* adalah sebagai berikut: Regresi, ANN (*Artificial Neural Network*), dan SVM (*Support Vector Machine*).

Permasalahan yang dapat diselesaikan dengan *supervised learning* adalah dengan algoritma yang sesuai, yakni :

- Mencari prediksi dari nilai riil (*continuous value*) digunakan pendekatan *regression problem*, yaitu : *linear* atau *multivariate regression*. *Multivariate regression* dapat digunakan untuk nilai prediksi dengan parameter atau *feature* yang majemuk.
- Mencari prediksi dari bilangan bulat digunakan pendekatan *classification problem*, yaitu : *logistic regression*. *Logistic regression* dapat digunakan untuk memilih benar atau salah ataupun *multivariate logistic regression* untuk klasifikasi yang terdiri dari lebih dari 2 kelas.

## 2. Pembelajaran Tak Terarah (*Unsupervised Learning*)

*Unsupervised learning* adalah metode tanpa adanya latihan (*training*). Teknik ini mempelajari bagaimana sebuah sistem dapat merepresentasikan pola *input* dengan cara merepresentasikan pola *input* dengan menggambarkan struktur statistikal terhadap keseluruhan pola *input*. *Unsupervised learning* ini sangat berbeda dengan *supervised learning* karena *unsupervised learning* tidak memiliki *target output* yang eksplisit atau tidak ada pengklasifikasian *input*. *Unsupervised learning* menjadi esensial karena di dalam teknik ini tidak ada informasi dari contoh yang tersedia. *Unsupervised learning* dapat digunakan untuk mencari struktur dari data yang ada dan data tersebut dikelompokkan berdasarkan informasi yang dimiliki metode *unsupervised learning* antara lain : *Clustering* dan *SOM (Self Organizing Map)*.

## 3. Pembelajaran Semi Terarah (*Semi-supervised Learning*)

*Semi-supervised learning* merupakan penggabungan antara pembelajaran *supervised* dan *unsupervised* dalam menghasilkan suatu fungsi dan memberikan pola *input-output* yang sesuai dengan jumlah data yang sedikit.

## 4. *Reinforcement Learning*

*Reinforcement learning* merupakan proses yang dibangun dari proses *mapping* untuk mendapatkan *reward* yang maksimal.

Prinsip yang digunakan dalam metode ini didasarkan pada teori *reinforcement* yang pada intinya adalah ‘konsekuensi mempengaruhi tindakan’. Ada 3 prinsip dasar pada teori *reinforcement* yakni :

- Konsekuensi yang berakibat baik mendorong terjadinya tindakan.
- Konsekuensi yang berakibat buruk mendorong berkurangnya tindakan.
- Konsekuensi yang tidak ada dampaknya tidak memperngaruhi tindakan.

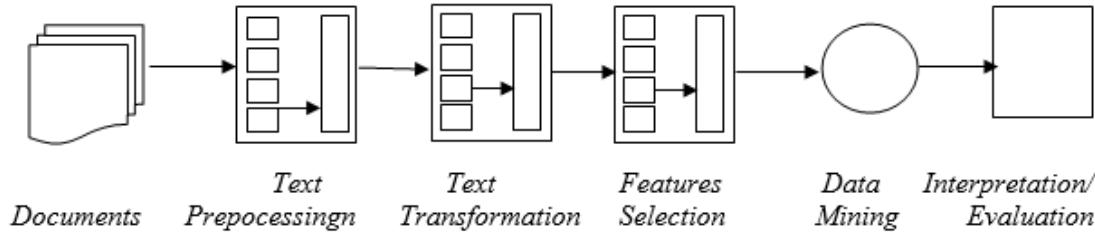
*Machine learning* memiliki dampak positif bagi para praktisi teknologi dalam mengembangkan teknologi-teknologi untuk membantu aktivitas tertentu. Salah satu dampak positif *machine learning* adalah pengecekan ejaan untuk tiap bahasa pada *microsoft word* yang hanya membutuhkan waktu yang singkat dalam proses pengecekannya karena pengecekan yang dilakukan secara manual akan membutuhkan

waktu yang sangat lama dan membutuhkan banyak tenaga untuk mendapatkan hasil penulisan yang lebih baik.

## 2.4 Text Mining

*Text mining* merupakan metode pengolahan data yang merupakan variasi dari *data mining* yang berusaha menemukan pola yang menarik dari sekumpulan data tekstual yang berjumlah besar (Kurniawan, et al., 2012). Selain klasifikasi, *text mining* juga digunakan untuk menangani masalah *clustering*, *information extraction*, dan *information retrieval* (Berry & Kogan, 2010).

Berikut proses *text mining*:



Gambar 2. Proses *text mining*

### 2.4.1 N-Grams

Menurut Russell dan Norvig (2010), *n-grams* didefinisikan sebagai urutan panjang simbol yang tertulis dengan sebutan unigram untuk 1-gram, bigram untuk 2-gram, trigram untuk 3-gram, dan seterusnya. *n-grams* merupakan sebuah potongan n karakter ataupun potongan n kata yang terdapat pada suatu *string* dalam kalimat tertentu. Misalnya dalam kata “ESAI” akan terdapat *n-grams* sebagai berikut:

Tabel 1. *N-grams* berbasis karakter

Nama	<i>n-grams</i> karakter
Uni-gram	E, S, A, I
Bi-gram	E, ES, SA, AI, I
Tri-gram	ES, ESA, SAI, EI, I
Quad-gram	ESA, SAI, AI, I
Dst...	

Karakter “\_” digunakan untuk merepresentasikan spasi di depan dan akhir kata. Pada tabel 1 dijelaskan *n-grams* berbasis karakter. Pada tabel 2 di bawah ini akan dijelaskan *n-grams* berbasis kata.

Tabel 2. *N-grams* berbasis kata

Nama	<i>n-grams</i> kata
<i>Uni-gram</i>	<i>n-grams</i> , merupakan sebuah, potongan, n, karakter, ataupun, potongan, n, kata, yang, terdapat, pada, suatu, <i>string</i> , dalam, kalimat, tertentu
<i>Bi-gram</i>	<i>n-grams</i> merupakan, merupakan sebuah, sebuah potongan, potongan n, n karakter, karakter ataupun, ataupun potongan, potongan n, n kata, kata yang, yang terdapat, terdapat pada, pada suatu, suatu <i>string</i> , <i>string</i> dalam, dalam kalimat, kalimat tertentu
<i>Tri-gram</i>	<i>n-grams</i> merupakan sebuah, merupakan sebuah potongan, sebuah potongan n, potongan n karakter, n karakter ataupun, karakter ataupun potongan, ataupun potongan n, potongan n kata, n kata yang, kata yang terdapat, yang terdapat pada, terdapat pada suatu, pada suatu <i>string</i> , suatu <i>string</i> dalam, <i>string</i> dalam kalimat, dalam kalimat tertentu tertentu
<i>Quad-gram</i>	<i>n-grams</i> merupakan sebuah potongan, merupakan sebuah potongan n, sebuah potongan n karakter, potongan n karakter ataupun, n karakter ataupun potongan, karakter ataupun potongan n, ataupun potongan n kata, potongan n kata yang, n kata yang terdapat, kata yang terdapat pada, yang terdapat pada suatu, terdapat pada suatu <i>string</i> , pada suatu <i>string</i> dalam, suatu <i>string</i> dalam kalimat, <i>string</i> dalam kalimat tertentu
Dst...	

## 2.5 Linear Regression

Regresi *linear* adalah metode statistika yang digunakan untuk membentuk model hubungan antara variabel terikat (dependen; respon; Y) dengan satu atau lebih variabel bebas (independen, prediktor, X) (Kurniawan, 2008). Jika jumlah variabel bebas hanya ada satu disebut regresi *linear* sederhana, sedangkan jumlah variabel bebas lebih dari satu disebut *multiple/multivariate regression*. Variabel yang mempengaruhi sering disebut variabel bebas, variabel independen atau variabel penjelas. Variabel yang dipengaruhi sering disebut dengan variabel terikat atau variabel dependen. Regresi *linear* hanya dapat digunakan pada skala interval dan ratio.

### Rumus :

$$Y = a + bX$$

Ket : Y = Variabel terikat

a = Nilai *intercept* (konstanta)

b = Koefisien regresi

X = Variabel bebas

### 2.6 Logistic Regression Model

Regresi *Logistic* merupakan salah satu bagian dari analisis regresi yang digunakan untuk memprediksi probabilitas kejadian suatu peristiwa, dengan mencocokkan data pada fungsi kurva *logistic* (Ng, 2002). Metode ini merupakan model *linear* umum yang digunakan untuk regresi binomial.

*Logistic regression model* membentuk persamaan atau fungsi dengan pendekatan *maximum likelihood* dalam memaksimalkan peluang pengklasifikasian objek yang diamati menjadi kategori yang sesuai dan kemudian mengubahnya menjadi koefisien regresi yang sederhana(Alexopoulos, ).

### 2.7 Multivariate Linear Regression

*Multivariate regression* adalah teknik yang memperkirakan sebuah *single regression model* dengan variabel bebas jamak. *Linear regression* dengan *multiple variables* juga dikenal sebagai *multivariate linear regression* (Ng, 2002).

### 2.8 Root Mean Square Error (RMSE)

*Root mean square error* sering digunakan sebagai ukuran perbedaan antara nilai prediksi model dan nilai asli yang diamati dari lingkungan yang dimodelkan. Perbedaan data individu juga disebut *residuals*, dan RMSE digunakan untuk menjumlahkan perbedaan individu ke dalam sebuah *single measure* dari *predictive power*.

Berikut merupakan rumus yang digunakan pada perhitungan RMSE (Chai & Draxler, 2012):

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n e_i^2}$$

Keterangan :

$n$  = Jumlah data

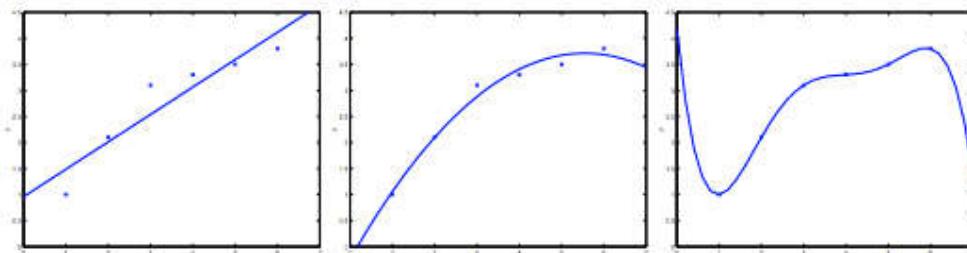
**Error! Reference source not found.** = Hasil kuadrat dari selisih absolut *predicted* dan *real value*.

## 2.9 Quadratic Weighted Kappa

*Quadratic weighted kappa* merupakan *error metric* atau merupakan tolak ukur akurasi prediksi suatu model antara dua buah prediktor, yakni sebuah model *machine learning*, dan model prediktif manual (Prediktif dengan campur tangan manusia) (Kaggle, 2012).

## 2.10 Overfitting dan Underfitting

*Overfitting* merupakan permasalahan akurasi prediksi  $y$  dari  $x \in R$ . Gambar 3 menunjukkan hasil *fitting* sebuah  $y = \theta_0 + \theta_1 x$  dalam sebuah dataset. *Overfitting* merupakan permasalahan dimana model mengalami *over-fit* dan tidak menjadi *general* seperti pada diagram paling kanan dari gambar berikut :



Gambar 3. *Overfitting* dan *underfitting*

Sedangkan *underfitting* merupakan kondisi dimana model prediktif gagal melakukan *fit* pada dataset seperti pada diagram paling kiri dari gambar diatas. Model yang baik adalah model yang tidak *overfitting* dan tidak *underfitting* sehingga model tersebut bersifat *general* dan memiliki akurasi yang seimbang pada *training*, *cross validate*, dan *test set*.

Berdasarkan gambar diatas dapat disimpulkan bahwa gambar yang sebelah kiri menunjukkan sebuah contoh *underfitting* dan gambar sebelah kanan menunjukkan contoh *overfitting*.

Ada beberapa opsi yang dapat dilakukan untuk mengatasi masalah *overfitting* diantaranya yaitu :

- Mengurangi jumlah *feature*

Mengurangi jumlah *feature* dapat kita lakukan dengan cara memilih *feature* yang akan digunakan dan penggunaan algoritma pemilihan model algoritma.

- Regularisasi

Regularisasi dapat dilakukan dengan cara menambahkan parameter *penalty* pada model

- *Cross Validation*

Sedangkan *underfitting* biasanya disebabkan kesalahan pemilihan model, banyak *feature* yang tidak relevan, dan masalah jumlah proporsi data. Oleh sebab itu, *underfitting* dapat diatasi dengan melakukan analisis pada penyebab-penyebab tersebut.

## 2.11 Cross Validation

*Cross validation* adalah pengujian standar yang dilakukan untuk memprediksi *error rate*. Data *training* dibagi secara *random* ke dalam beberapa bagian dengan perbandingan yang sama kemudian *error rate* dihitung bagian demi bagian, selanjutnya hitung rata-rata seluruh *error rate* untuk mendapatkan *error rate* secara keseluruhan (Leidiyana, 2013).

## 2.12 Preprocessing

*Preprocessing* adalah tahap awal untuk meringkas *input* teks menjadi data yang siap diolah. Tahap yang terkandung dalam *preprocessing* adalah *case folding*, *tokenizing* dan *stemming*, dan *stop-word removal* (Mahana, et al., 2012).

*Case Folding* merupakan proses menyamakan setiap huruf dalam sebuah teks menjadi huruf besar ataupun huruf kecil(Launder, et al., 1990). *Tokenizing* merupakan proses memisahkan sebuah teks menjadi beberapa segmen yang dipisahkan berdasarkan tanda pemisah atau *token* tertentu (Launder, et al., 1990). *Stop-word removal* merupakan proses pengeliminasian kata henti dari teks(Launder, et al., 1990). *Stemming* merupakan proses untuk mendapatkan bentuk kata dasar dari setiap kata pada teks (Riandyanai, et al., 2014).

## 2.13 Features

*Features* merupakan kumpulan variabel bebas (*independent variable*) dalam model *machine learning*(Ng, 2002). Variable tersebut akan digunakan untuk menjadi penentu dari *predicted output* (*dependent variable*) dengan pemodelan tertentu.

## 2.14 Feature Extraction

*Features extraction* merupakan proses pengumpulan informasi yang menjadi dasar penilaian esai. *Features extraction* terdiri dari dua fase utama (Guyon & Eliseff, 2006):

- 1) Identifikasi dan konstruksi *feature*
- 2) Pemilihan *feature* yang relevan

Pengetahuan tentang bahasa dan penulisan sangat penting untuk mendefinisikan semua *feature* yang mungkin relevan untuk mengevaluasi esai.

*Feature* yang berbeda dapat dikombinasikan dengan cara yang berbeda untuk menciptakan *feature* baru yang mungkin mewakili karakteristik esai baru. Salah satu contohnya adalah menggabungkan *parts speech counter* ke dalam *feature* yang disebut *Nominal Ratio* yang mengukur kerapatan gagasan untuk teks.

*Feature extraction* melibatkan sumber daya yang dalam menjelaskan sebuah dataset secara akurat. Dalam proses melakukan analisis sata yang sangat kompleks, salah satu masalah utama berasal dari jumlah variabel yang terlibat analisis dengan sejumlah variabel biasanya membutuhkan sejumlah memori yang besar dan daya komputasi atau algoritma klarifikasi yang *overfit*.

*Feature extraction* adalah fase penting dalam identifikasi karena setiap huruf mempunyai keunikan tersendiri sehingga membedakan suatu huruf tersebut dari huruf yang lain. *Feature extraction* bertujuan untuk mendapatkan karakteristik suatu karakter yang membedakannya dari karakter lain, yang disebut *feature*. Karakteristik ini dapat berupa titik koordinat awal dan akhir dari suatu goresan dan fragmen garis penyusun karakter.

*Feature* yang diperoleh digunakan untuk mendefinisikan kelas suatu karakter.

*Feature* merupakan *high-level-attribute* dari data goresan karakter. Setelah proses *feature extraction* selesai, maka didapat *feature* dari sebuah huruf. *Feature* ini mempresentasikan informasi struktural dari sebuah huruf.

## 2.15 Features Selection

*Variable selection* atau *features selection* adalah proses seleksi *feature* di dalam data untuk mendapatkan *features* dengan kontribusi yang paling relevan dengan masalah pemodelan prediktif (Guyon & Eliseff, 2003). Secara umum, *features selection* merupakan proses memilih subset *features* yang relevan dalam membangun sebuah model *machine learning*.

*Features selection* berbeda dari *dimensionally reduction*. Namun, kedua metode tersebut sama-sama bekerja untuk mengurangi jumlah *features* dalam dataset, tetapi *dimensionally reduction* melakukannya dengan menciptakan kombinasi baru dari *features*, sedangkan *features selection* menambah dan mengeliminasi *features* di dalam data tanpa melakukan perubahan(Guyon & Eliseff, 2003).

Contoh metode *dimensionality reduction* meliputi *principal component analysis*, *singular value decomposition* dan *Sammon's mapping*.

*Features selection* itu sendiri sebagian besarnya bertindak sebagai *filter* untuk menghilangkan *features* yang tersedia.

### 2.15.1 Kegunaan Features Selection

Kegunaan dari *features selection* antara lain (Guyon & Eliseff, 2003):

- Membantu dalam membuat model prediksi yang akurat dengan memilih *features* yang akan memberikan akurasi yang baik ataupun lebih baik, meskipun hanya tersedia sedikit data.
- Dapat digunakan untuk mengidentifikasi dan menghapus *features* yang tidak diperlukan, tidak relevan dan berlebihan dari data yang tidak memberikan kontribusi pada akurasi model prediktif.
- Lebih sedikit *features* diinginkan karena mengurangi kompleksitas model, dan model lebih sederhana, dan lebih mudah dipahami dan dijelaskan.
- Meningkatkan kinerja prediksi dan menyediakan komputasi yang lebih cepat dan lebih hemat.
- Untuk menyederhanakan model prediktif agar lebih mudah dipahami.

## 2.15.2 Algoritma Features Selection

Ada tiga kelas umum algoritma *features selection* yaitu *filter methods*, *wrapper methods* dan *embedded methods* (Guyon & Eliseff, 2003).

- *Filter Method*

*Filter method* dari *features selection* menerapkan ukuran statistik dalam penetapan *scoring* dari masing-masing *features* tersebut mengurutkan berdasarkan peringkat skor kemudian dipilih untuk disimpan atau dikeluarkan dari *database*. Metode-metode yang digunakan sering *univariate* dan mempertimbangkan *features* tersebut secara mandiri, atau berkaitan dengan variabel yang dependen. Beberapa contoh dari *filter method* meliputi *chi squared test*, *information gain* dan *correlation coefficient scores*.

- *Wrapper Method*

*Wrapper method* mempertimbangkan pemilihan dari sebuah *set features* sebagai masalah pencarian, dimana kombinasi yang berbeda disusun, dievaluasi dan dibandingkan dengan kombinasi lainnya. Sebuah model prediksi digunakan untuk mengevaluasi kombinasi *features* dan menetapkan nilai berdasarkan akurasi model.

Proses pencarian mungkin sistematis seperti pencarian terbaik, stokastik seperti algoritma *random hill-climbing*, atau menggunakan metode heuristik, *forward selection* dan *backward elimination*.

- *Embedded Method*

*Embedded method* yaitu metode yang belajar dari *features* yang memiliki kontribusi terbaik dalam akurasi model saat model sedang dibuat. Jenis yang paling umum dari pemilihan *features embedded method* adalah metode regularisasi. Metode regularisasi juga disebut metode hukuman yang memperkenalkan variabel *penalty* ke dalam optimalisasi algoritma prediktif (seperti algoritma regresi) yaitu model yang mengarah ke kompleksitas yang lebih rendah (koefisien yang lebih sedikit).

Contoh algoritma regularisasi adalah LASSO, *elastic net* dan *ridge regression*.

## 2.16 Features Scaling

*Feature scaling* adalah metode yang digunakan untuk standarisasi berbagai variabel independen atau *feature* data dengan rentang yang bervariasi(Ng, 2002). Dalam data processing juga dikenal dengan istilah normalisasi data dan umumnya dilakukan pada tahapan *preprocessing*.

*Features scaling* juga direkomendasikan secara luas dan banyak dilakukan dalam *machine learning*. *Feature scaling* sangat penting dalam konteks rentang nilai atau *raw data* yang menyebabkan banyak algoritma *machine learning* gagal, hal ini diakibatkan karena mayoritas perhitungan pada *machine learning* melibatkan penghitungan jarak antara dua titik (*range calculation*).

## 2.17 Current Research

Pada penelitian sebelumnya digunakan model *regularized linear regression* dengan *5-fold cross validation*, dan tingkat akurasi prediksi model pada essai set 1 (narasi, persuasi, ekspositori) adalah sebesar 0.80 dengan *metric quadratic weighted kappa* (Mahana, et al., 2012).

## 2.18 Ringkasan

Setelah melakukan pembelajaran terhadap bahan pustaka yang telah ditentukan, maka diputuskan untuk memilih model *multivariate regression* tipe *linear* dan *logistic*, dan metode *wrapper* dalam analisis penilaian esai otomatis dengan menggunakan *root mean square error*(RMSE) serta *Quadratic Weighted Kappa* untuk memperhitungkan tingkat akurasi dari setiap esai yang akan diberikan penilaian. Kemudian setelah model *final* didapat maka akan dilakukan penambahan *n-grams feature* yakni *bigram* dan *trigram*.

## Bab III Analisis

### 3.1 Analisis Data

Data untuk penggeraan Tugas Akhir ini bersumber dari *Kaggle.com*, oleh William dan Flora Hewlett Foundation (Kaggle, 2012). Data tersebut terdiri dari 8 dataset dengan karakteristik beragam, dimana untuk penelitian ini akan digunakan *essay set 1* yang merupakan *essay* dengan jenis, narasi, persuasi, dan ekspositori.

*Dataset* merupakan data dari kompetisi menulis *essay* yang diadakan oleh William dan Flora Hewlett Foundation (Hewlett) untuk sekolah menengah dengan tema “*Opinion on the Effects Computers Have on People*”.

*Dataset* tersebut di unduh dalam bentuk *Keyed JSON* dan akan dikonversi kedalam format *excel* dengan tujuan untuk mempermudah pembacaan data dengan adanya bantuan *library Python 2.7* yaitu *xlrd* (Machine, 1990). *Library* tersebut merupakan *tools* untuk mempermudah proses pembacaan(*read*) dan penulisan (*write*) pada *file excel*. Sedangkan untuk proses konversi format dataset digunakan *online conversion tool* (Orchard, 2017). Berikut ini adalah contoh *file JSON* dari dataset:

```
[  
  {  
    "essay_id": 1,  
    "essay_set": 1,  
    "essay": "Dear local newspaper, I think effects computers have on people are great learning skills/affects because they give us time to chat with friends/new people, helps us learn about the globe(astronomy) and keeps us out of trouble! Thing about! Don't you think so? How would you feel if your teenager is always on the phone with friends! Do you ever time to chat with your friends or business partner about things. Well now - there's a new way to chat the computer, there's plenty of sites on the internet to do so: @ORGANIZATION1, @ORGANIZATION2, @CAPS1, facebook, myspace etc. Just think now while you're setting up meeting with your boss on the computer, your teenager is having fun on the phone not rushing to get off cause you want to use it. How did you learn about other countries/states outside of yours? Well I have by computer/internet, it's a new
```

way to learn about what going on in our time! You might think your child spends a lot of time on the computer, but ask them so question about the economy, sea floor spreading or even about the @DATE1's you'll be surprise at how much he/she knows. Believe it or not the computer is much interesting then in class all day reading out of books. If your child is home on your computer or at a local library, it's better than being out with friends being fresh, or being perpressured to doing something they know isn't right. You might not know where your child is, @CAPS2 forbidded in a hospital bed because of a drive-by. Rather than your child on the computer learning, chatting or just playing games, safe and sound in your home or community place. Now I hope you have reached a point to understand and agree with me, because computers can have great effects on you or child because it gives us time to chat with friends/new people, helps us learn about the globe and believe or not keeps us out of trouble. Thank you for *listening.*",

```
"rater1_domain1": 4,  
"rater2_domain1": 4,  
"rater3_domain1": "",  
"domain1_score": 8,  
"rater1_domain2": "",  
"rater2_domain2": "x",  
"domain2_score": "x",  
"rater1_trait1": "x",  
"rater1_trait2": "x",  
"rater1_trait3": "x",  
"rater1_trait4": "x",  
"rater1_trait5": "x",  
"rater1_trait6": "x",  
"rater2_trait1": "x",  
"rater2_trait2": "x",  
"rater2_trait3": "x",
```

```
"rater2_trait4": "x",
"rater2_trait5": "x",
"rater2_trait6": "x",
"rater3_trait1": "x",
"rater3_trait2": "x",
"rater3_trait3": "x",
"rater3_trait4": "x",
"rater3_trait5": "x",
"rater3_trait6": "x",
}
]
```

Dapat dilihat bahwa data tersebut mengandung beberapa *key* yang tidak diperlukan untuk melakukan *training* pada model regresi yang akan dilakukan. Adapun *key* yang akan digunakan untuk proses *training* pada saat eksperimen akan dilakukan adalah sebagai berikut:

- “*essay\_id*” : Merupakan id (*primary key*) yang menjadi *features* unik dari sebuah esai.
- “*essay*” : Merupakan isi dari esai itu sendiri.
- “*rater1\_domain1*” : Hasil penilaian dari juri pertama.
- “*rater1\_domain2*” : Hasil penilaian dari juri kedua.
- “*domain1\_score*” : Nilai *final* yang merupakan nilai yang berasal hasil dari penjumlahan nilai dari kedua juri.

### **3.2 Prosedur dan Standar Penilaian Manual**

Adapun *standard* dan prosedur penilaian esai secara manual terhadap dataset adalah sebagai berikut(Kaggle, 2012):

#### **3.2.1 Prompt(Kata Pengantar)**

Komputer merupakan teknologi yang sedang menjamur di dalam kehidupan sehari-hari kita, hampir setiap sisi kehidupan telah melibatkan komputer. Pada saat ini terdapat beberapa pro dan kontra tentang pemanfaatan ataupun manfaat dari komputer itu sendiri terhadap sisi kehidupan kita sebagai manusia.

#### **3.2.2 Instruction(Instruksi)**

Tulisan esai tentang manfaat komputer terhadap kehidupan orang banyak, dan mengajak pembaca untuk setuju dengan argumen yang telah dibuat.

#### **3.2.3 Rubic Guidelines(Standar Penilaian)**

Pada sub bab ini terdapat penjelasan *score point* 1 sampai *score point* 6.

##### **Score Point 1:**

Esai tidak mendapat dukungan yang baik, hanya berupa dukungan minim. Adapun kriteria khusus untuk esai dengan skor 1, adalah sebagai berikut:

- Berisi beberapa rincian yang kabur/tidak jelas
- Canggung dan terfragmentasi
- Sulit dimengerti
- Tidak menarik perhatian pembaca.

##### **Score Point 2:**

Respon yang memberatkan dari pembaca yang mungkin atau tidak mengambil perhatian si pembaca. Adapun kriteria khusus untuk esai dengan skor 2, adalah sebagai berikut:

- Belum terstruktur dengan baik
- Hanya berisi alasan secara umum dan tidak disertai rincian-rincian yang detail
- Canggung atau terlalu sederhana
- Hanya menarik sedikit perhatian pembaca.

### **Score Point 3:**

Mendapat respon minimal dari pembaca, tetapi masih kurang cukup argumen dukungan dan rincian. Adapun kriteria khusus untuk esai dengan skor 3, adalah sebagai berikut:

- Memiliki alasan dengan elaborasi minimal dan lebih umum tanpa rincian spesifik
- Sudah mulai terstruktur dengan baik
- Masih terdapat beberapa kerancuan pada beberapa kata/kalimat peralihan.
- Sedikit menarik perhatian pembaca.

### **Score Point 4:**

## **institut teknologi**

Sudah terdapat argumen yang mendukung dengan pernyataan yang dikemukakan. Adapun kriteria khusus untuk esai dengan skor 4, adalah sebagai berikut:

- Alasan yang memadai telah diuraikan dengan campuran umum dan rincian spesifik
- Pengorganisasian kalimat yang memuaskan
- Transisi kalimat yang lancar dan koheren
- Menarik perhatian pembaca.

### **Score Point 5:**

Mendapat perhatian yang baik dari pembaca, dimana argumen sudah dikemukakan dengan jelas, disertai bukti pendukung yang spesifik dan rinci. Adapun kriteria khusus untuk esai dengan skor 5, adalah sebagai berikut:

- Sudah terstruktur dengan baik dan rapi
- Argumen didukung dengan alasan yang kuat, deskriptif dan rinci
- Transisi dalam esai sudah mengikuti aturan-aturan yang berlaku
- Cukup banyak menarik perhatian pembaca.

## Score Point 6:

Esai ditulis dengan struktur yang sangat baik, dan mendapat respon yang baik dari pembaca dikarenakan unsur persuasif dalam esai terstruktur dengan baik dan tepat sasaran. Adapun kriteria khusus untuk esai dengan skor 6, adalah sebagai berikut:

- Sangat terstruktur dengan baik
- Transisi dalam kalimat sudah koheren, sesuai dengan tata bahasa dan mengalir dengan baik
- Tata kata, pemilihan kata, tata kalimat, dan tata paragraf sudah mengikuti aturan-aturan yang tersedia

Memberi kesan persuasif yang kuat pada pembaca.

### 3.2.4 Adjudication Rules(Penentuan Skor Final)

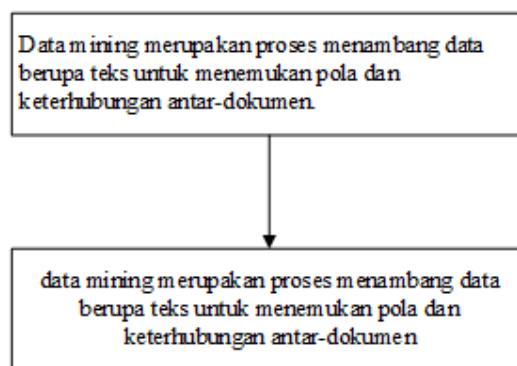
Skor *final* merupakan hasil penjumlahan dari skor juri 1 dan skor juri 2.

## 3.3 Analisis Preprocessing

Tahap awal yang dilakukan adalah *preprocessing* dokumen, dimana *output*-nya adalah data yang sudah bersih, sehingga *feature extraction* dapat dilakukan. *Text preprocessing* pada Tugas Akhir ini terdiri atas tiga tahapan umum yaitu tahap *case folding*, *tokenization*, dan *stopwords removal*.

### 3.3.1 Case Folding

Tahap *case folding* adalah proses mengubah seluruh karakter pada esai menjadi huruf kecil. Tahapan ini membantu proses *features extraction* agar huruf pada dataset menjadi seragam. Contoh proses *case folding* dapat dilihat pada gambar 4 dibawah ini:

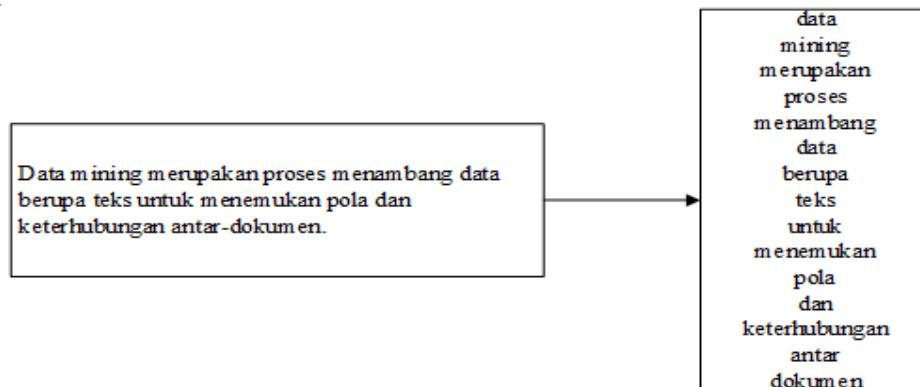


Gambar 4. Proses *case folding*

Pada skema diatas dapat dilihat bahwa *input* karakter “*Data mining*” mengalami perubahan yaitu karakter menjadi huruf kecil menjadi *data mining*. Proses *case folding* dilakukan secara *iterative* pada *input* dimana setiap karakter di cek dan dikonversi menjadi huruf kecil.

### 3.3.2 Tokenization

Tahap *tokenization* adalah proses memisahkan teks dalam esai pada dataset ke dalam token-token berdasarkan token pemisah. Contoh proses *tokenizing* adalah dengan tanda spasi sebagai token pemisah dapat dilihat pada gambar 4 di bawah ini :



Gambar 5. Proses *tokenizing* dengan tanda spasi sebagai token pemisah

Pada gambar diatas dapat dilihat bahwa *input* masih berupa sebuah *string* yang utuh, setelah dilakukannya *tokenization*, maka diperoleh *array* dari setiap kata yang ada pada kalimat *input*.

### 3.3.3 Stopword Removal

Tahapan ini bertujuan untuk penghapusan *stopword* seperti *the*, *of*, *is*, *at* dsb. Dengan dilakukannya tahapan ini maka pada saat dilakukannya *features extraction* hasil yang diberikan akan menjadi lebih maksimal dikarenakan kita dapat berfokus pada kata-kata yang menjadi inti dari esai tersebut. Akan tetapi ada beberapa *feature* yang membutuhkan *stopword* ini sehingga diperlukan data yang tidak mengalami tahapan *preprocessing* ini, dan akan dilakukan *features extraction* secara terpisah.

### 3.4 Analisis Feature Extraction

Pada tahapan ini dilakukan pengumpulan informasi yang menjadi dasar penilaian esai, dengan dilakukannya *features extraction* adapun *feature* yang akan diekstrak dikelompokkan menjadi 6 bagian yaitu(Mahana, et al., 2012):

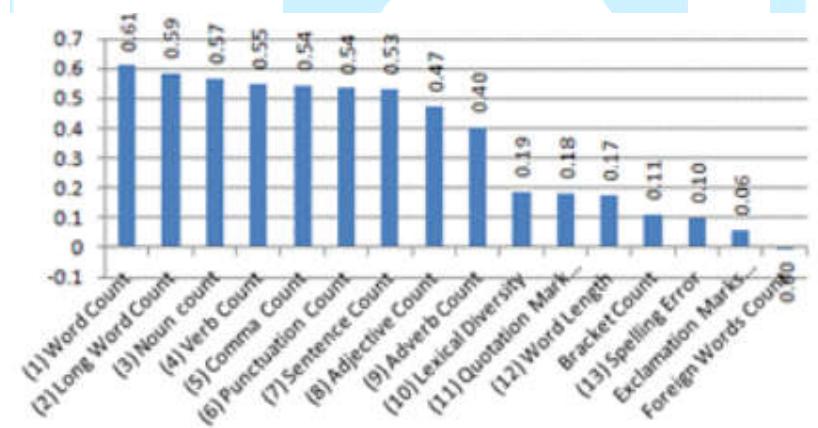
1. *BOW*: Bow digunakan sebagai sebuah basis untuk memilih *feature* (kata) yang baik dalam prediksi skor esai. Didalam *Library text mining* terdapat utilitas *TermDocumentMatrix* untuk membuat sebuah *bag* dari kata-kata, *bag* yang dimaksud merupakan sebuah ukuran keberadaan konten tertentu dan konsep-konsep dalam masing-masing dataset. Untuk setiap esai, dilakukan penghapusan *stop-word* seperti *the*, *of*, *is*, *at*, kemudian dilakukan perbandingan untuk frekuensi kata-kata dalam bahasa yang umum.
2. *Numerical Features*: *feature* seperti total jumlah kata per esai, rata-rata panjang kata per esai dan jumlah kalimat. Esai terlebih dahulu dilakukan *tokenizing* menggunakan *python*. Hasil dari *tokenizing* digunakan untuk menghitung jumlah kata, jumlah kalimat dan menghitung nilai-nilai karakter.
3. *Part of Speech count*: Berbagai macam perhitungan kata seperti kata benda (*noun*), kata keterangan (*adjective adverbs*), dan kata kerja(*verb*). *Feature* ini dapat digunakan sebagai perantara dasar untuk pemilihan kata dan menggunakan *python NLTK-part-of-speech count* untuk pemisahan masing masing jenis kata.
4. *Orthography*: Ilmu yang mempelajari tentang cara penulisan dan pengucapan huruf-huruf bahasa Inggris dengan benar. Dilakukan pengumpulan jumlah kesalahan ejaan per esai untuk menguji karakteristik dengan melakukan penghapusan tanda baca pada esai dan di *tokenized* terlebih dahulu. Yang dibutuhkan dalam *orthography* ini adalah PyEnchant 1.6.5 *spell checker*(pemeriksaan ejaan) dengan kamus aspell untuk mendapatkan hitungan *misspelt*(kesalahan ejaan) kata per esai.
5. *Structured and Organization*: pemberian tanda baca adalah indikator yang baik dari sebuah esai yang terstruktur dan terorganisir. Tanda baca yang berbasis *feature* diekstraksi menggunakan *reguler expression* yang sesuai dari esai setelah dilakukan pemisahan deret kata dalam kalimat.

6. *N-grams*: jumlah unik dari setiap *n-grams* akan dihitung, dan ditambahkan sebagai *feature*. *Feature* ini akan di ekstrak dengan menggunakan library *NLTK* pada *Python 2.7*.

### 3.5 Analisis Features Selection

Tidak semua *features* digunakan sebagai parameter penentu skor esai, untuk itu lah dilakukan tahapan ini sehingga pada saat dilakukannya proses *final training* untuk pembentukan model prediktor skor esai, akan didapatkan tingkat keakuratan yang maksimal.

Pada penelitian sebelumnya dilakukan *forward features selection*. Proses *forward selection* dimulai dengan melakukan proses *training* dengan satu *feature* untuk seluruh *features* yang tersedia, lalu tingkat akurasinya diukur dan dicatat. Kemudian semua *features* diurutkan berdasarkan tingkat akurasi dari yang paling baik hingga yang paling rendah seperti pada gambar berikut.



Gambar 6. Tingkat akurasi

Langkah selanjutnya yang akan dilakukan adalah proses *forward selection* itu sendiri, adapun proses tersebut dimulai dengan dilakukannya proses *training* dengan *feature* pertama yang memiliki tingkat akurasi tertinggi, kemudian tingkat akurasinya diukur. Selanjutnya akan ditambahkan *feature* kedua dan dilakukan proses *training* lagi dan tingkat akurasinya diukur dan dibandingkan dengan tingkat akurasi sebelum penambahan *feature* tersebut, apabila tingkat akurasinya bertambah dibandingkan dengan sebelumnya, maka *feature* tersebut akan dipertahankan, sebaliknya apabila tingkat akurasinya

berkurang, maka *feature* tersebut akan dieliminasi. Proses ini dilakukan secara iteratif sebanyak jumlah *feature* yang kita miliki.

Metode *forward selection* ini memiliki kelemahan dikarenakan tidak semua subset ataupun permutasi dari *feature* dievaluasi, sehingga besar kemungkinan *feature* dengan tingkat akurasi maksimum tidak dievaluasi. Oleh karena itu, pada penelitian ini akan dilakukan metode *wrapper* untuk *features selection*. Secara umum proses pada metode *wrapper* terbagi atas :

- Proses *generate* Permutasi / Subset dari *feature*
- Proses *training* untuk setiap permutasi dari *feature*
- Proses evaluasi untuk setiap permutasi dari *feature* untuk pengukuran tingkat akurasi

Berdasarkan analisis sekilas, tentu metode ini lebih efisien dibandingkan dengan *forward selection* karena metode ini menerapkan konsep *Brute-Force (Exhaustive Search)* yang mengevaluasi seluruh permutasi ataupun subset dari *feature* yang tersedia.

Tentunya *cost* atau biaya komputasi yang dibutuhkan untuk proses ini akan bertumbuh secara *eksponensial*. Akan tetapi karena jumlah dataset yang dimiliki masih tergolong relatif sedikit ±1800 data, maka hal ini masih memungkinkan untuk dilakukan. apabila data yang digunakan sudah tergolong data *massive* atau data dengan populasi yang cukup besar, dapat dilakukan teknik *sampling data* sehingga tidak semua data digunakan pada proses ini.

Dengan metode ini, setiap kombinasi dari *feature* akan dicatat dan diukur tingkat akurasinya, sehingga dapat dilakukan analisis lebih mendalam seperti korelasi antar 2 atau lebih *feature*. Misalnya terdapat *feature* luas dan panjang tanah dari sebuah rumah, maka ketika memprediksi harga rumah , kedua *feature* tersebut dapat direpresentasikan menjadi sebuah *feature* baru yakni luas tanah, sehingga dimensi dari dataset yang diberikan menjadi lebih sedikit dan tentunya model prediktif menjadi lebih efisien.

### 3.6 Analisis Feature Scaling

*Features scaling* dapat mempercepat *gradient descent* mencapai titik optimal lebih cepat (*convex*) dengan setiap *input features* yang rentang yang sama. Hal ini dikarenakan parameter  $\theta$  akan turun dengan cepat pada kurva optimum dengan rentang kecil. Sementara apabila rentang dari *input* tidak merata, maka fungsi optimum akan lambat dalam mencapai titik optimum (*convex*), bahkan tidak dapat mencapai titik optimal tersebut. Cara yang digunakan untuk mencegah hal ini adalah dengan memodifikasi rentang variabel *input* sehingga rentang variabel *input* menjadi sama dengan rentang sebagai berikut :

**institut teknologi**

Atau :  $-0.5 \leq x_{(i)} \leq 0.5$

Dua teknik untuk membantu *feature scaling* dan *mean normalization*. *Feature scaling* membagi nilai *input* dengan rentang nilai maksimum dikurangi dengan nilai minimum dari variabel *input* yang menghasilkan sebuah rentang baru. *Mean normalization* mengurangkan nilai rata-rata untuk variabel masukan nilai-nilai untuk variabel masukan sehingga menghasilkan nilai rata-rata baru untuk variabel *input* hanya nol.

Di bawah ini terdapat rumus penyesuaian *feature scaling* dengan *mean normalization* :

$$x_i := \frac{x_i - \mu_i}{s_i}$$

### 3.7 Model Evaluation

Pada sub bab *model evaluation* menjelaskan *root mean square error* dan *quadratic weighted kappa*. RMSE pada hakikatnya merupakan model *evaluation* untuk model prediktif jenis regresi. Sedangkan Kappa merupakan model evaluation untuk model prediktif tipe klasifikasi dan untuk evaluasi keseluruhan model prediktif(*logistic* dan *linear*), maka akan digunakan kedua *error metric* tersebut pada kedua model prediktif (*logistic* dan *linear*) agar tercapai *model evaluation* yang seimbang.

#### 3.7.1 RMSE

Untuk setiap model yang didapatkan melalui proses *training*, baik untuk kepentingan *features selection (temporary-model)* ataupun model *final*, akan dilakukan evaluasi untuk mendapatkan tingkat akurasi prediksi model. Adapun pengukuran dilakukan dengan

menggunakan *root mean square error* atau (RMSE) yang telah dijelaskan sebelumnya pada sub bab 2.3.2. Adapun proses perhitungan didasari dengan rumus berikut:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n e_i^2}$$

Sebagai contoh untuk data berikut dimana data hasil prediksi dari model sudah dicatat,

Tabel 3. Hasil prediksi

real value	predicted value
7	8
8	7
5	7
6	8
8	4
7	6

Maka dapat disimpulkan bahwa  $n = 6$  yang menyatakan jumlah data dan  $e$  dapat dihitung dengan mencari nilai absolut dari selisih *real\_value* dan *predicted\_value* atau dapat dinotasikan dengan:

$$e_i = | \text{real\_value}_i - \text{predicted\_value}_i |$$

Sehingga akan didapat hasil untuk  $e_i$  dan  $e_i^2$  seperti berikut :

Tabel 4. Hitung nilai e

real_value	predicted_value	$e_i$	$e_i^2$
7	8	1	1
8	7	1	1
5	7	2	4
6	8	2	4
8	4	4	16
7	6	1	1

Selanjutnya akan dihitung total jumlah (*sigma*) dari  $e_i^2$  yakni dengan nilai sebesar 27, kemudian dilakukan perhitungan *rata-rata* (*mean*) dan *akar*(*root*) sehingga nilai akhir dari RMSE = 2.1321.

Dapat disimpulkan bahwa RMSE merupakan fungsi yang harus di *minimize*, dimana rentang nilainya ada pada rentang 0 – tak hingga. Semakin mendekati 0 maka akurasi model prediktif semakin baik. Proses pengukuran RMSE dilakukan dengan bantuan *package scikit-learn*.

### 3.7.2 Quadratic Weighted Kappa

*Quadratic Weighted Kappa* digunakan sebagai salah satu *metric/alat ukur* akurasi prediktif model dalam penelitian penilaian esai otomatis. Kappa mengukur tingkat akurasi persetujuan antara model *machine learning* dan *score* esai dari manusia. Rentang nilai *Quadratic Weighted Kappa* berkisar antara 0 dan 1, dimana nilai 1 berarti akurasi 100% sedangkan nilai 0 berarti akurasi 0%. Jadi dapat disimpulkan bahwa *Quadratic Weighted Kappa* merupakan fungsi yang harus di *maximize* dengan batas atas adalah sama dengan 1. Semakin dekat nilai Kappa dengan 1 maka model prediktif semakin akurat. Untuk formula dari *Quadratic Weighted Kappa* adalah sebagai berikut :

$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}}$$

W merupakan *squared difference* dari data ke-i dan ke-j dan E merupakan *frequency matriks* untuk kalkulasi Kappa, da O<sub>i,j</sub> merupakan banyaknya kemunculan skor i hasil dari grader 1 dan j hasil dari grader 2 pada *Kaggle Autograding System*(Kaggle, 2012).

Pada penelitian ini, penulis akan menggunakan fungsi *Kappa* pada *package python* yakni *scikit-learn*, dikarenakan nilai dari E pada *grader Kaggle* menggunakan bawaan *scikit-learn*. *Scikit-learn* adalah modul *Python* yang mengintegrasikan berbagai algoritma *machine learning* untuk *medium-scale supervised* dan *unsupervised learning*. Kedua model prediktif pada penelitian ini akan diukur berdasarkan nilai *Quadratic Weighted Kappa* dengan dilakukannya *10-fold cross validation*.

### 3.8 Analisis Training Model

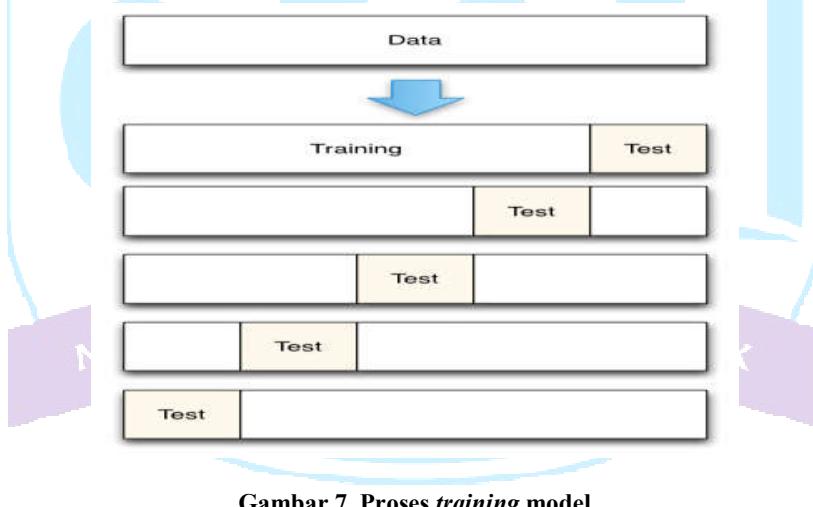
Pada sub bab analisis *training model* terdapat dua model yaitu regresi *linear* dan *logistic regression linear*.

#### 3.8.1 Regresi Linear

Analisis Regresi *Linear* Sederhana digunakan untuk mengukur rasio pengaruh antara satu variabel prediktor(variabel bebas) terhadap variabel terikat (DIKLI, 2006).

Tahapan ini merupakan tahapan penting, dikarenakan hasil evaluasi ataupun tingkat akurasi dari proses ini dilakukan pada saat penarikan kesimpulan dan validasi hipotesa. Model yang digunakan pada saat proses adalah *multivariate regression* dengan *features* hasil dari proses *features selection*. Untuk proses *training*, akan digunakan *package scikit-learn* dan beberapa *package* pendukung *Python 2.7* lainnya seperti *numpy*, *scipy*, *pandas*, dan *xlrd*.

Untuk setiap proses *training* akan dilakukan *10-fold-cross-validation* sehingga komposisi data yang dapat dilihat pada diagram berikut:



Gambar 7. Proses *training* model

Dapat dilihat bahwa setiap data terdistribusi secara merata untuk pembagian *training* dan *test set*, untuk langkah selanjutnya dilakukan pula proses *training* dengan penambahan parameter *n-grams*.

Berikut merupakan pemodelan matematis dari *multivariate linear regression*:

Dibawah ini terdapat notasi persamaan yang dapat dimiliki dari sejumlah variabel *input* :

$x_j^{(i)}$  = nilai dari features  $j$  dalam Error! Reference source not found.

$x^{(i)}$  = kolom vektor dari semua *features*

M = jumlah data *training*

N = |Error! Reference source not found.|; (nomor features)

Fungsi hipotesis dalam bentuk *multivariable* fungsi yang dapat dimodelkan dengan persamaan berikut :

Error! Reference source not found.(Error! Reference source not found.) = Error!

Reference source not found. Error! Reference source not found. | Error! Reference source not found. | Error! Reference source not found.

Reference source not found.2Error! Reference source not found.2 Error! Reference source not found.3Error! Reference source not found.3 **Error! Reference source not found.** Error! Reference source not found.nError! Reference source not found.n

Apabila **Error! Reference source not found.** didefinisikan sebagai matriks dari *cost function*, dan **Error! Reference source not found.** merupakan matriks dari *feature*, dan **Error! Reference source not found.** maka fungsi hipotesis, ataupun model prediktif dapat direpresentasikan dalam perkalian matriks sebagai berikut:

Error! Reference source not found.(Error! Reference source not found.) =Error! Reference source not found. found. Error! Reference source not found. found. Error! Reference source not found. found. Error! Reference source not found.] Error! Reference source not found.n ] Error! Reference source not found. = Error! Reference source not found.

Dimana **Error! Reference source not found.** merupakan transpose dari matriks **Error! Reference source not found.**. Notasi ini mewakili satu hipotesa dari dataset.

Apabila hipotesa direpresentasikan dalam  $X$  *row-wise*, maka akan didapatkan notasi-notasi berikut :

$$x_0^{(i)} = 1 :$$

Error! Reference source not found. = Error! Reference source not found. , Error! Reference source not found. = Error! Reference source not found.

Hasil kalkulasi hipotesis sebagai kolom vektor dari ukuran ( $m \times 1$ ) adalah sebagai berikut:

$h_{\theta}(X) = X\theta$

# institut teknologi

**Repeat{**

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[ \left( \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] \quad j \in \{1, 2, \dots, n\}$$

**}**

Pada saat **Error! Reference source not found.**  $\theta_j$  melakukan regularisasi. Beberapa pembaharuan saturan manipulasi juga dapat direpresentasikan sebagai berikut:

$$\theta_j := \theta_j \left( 1 - \alpha \frac{\lambda}{m} \right) - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Syarat pertama dalam persamaan diatas,  $1 - \alpha$  **Error! Reference source not found.** akan selalu lebih kecil dari 1. Secara intuitif mengurangi nilai  $\theta_j$  oleh beberapa bilangan setiap melakukan pembaharuan.

- **Normal Equation dengan Regularisasi**

Pendekatan regularisasi menggunakan metode alternatif *non-iterative* persamaan normal. Dalam menambahkan regularisasi, persamaan adalah sama seperti persamaan sebelumnya, kecuali jika menambahkan istilah lain dalam kurung.

Fungsi yang digunakan dalam *normal equation* adalah sebagai berikut:

$$\theta = (X^T X + \lambda \cdot L)^{-1} X^T y$$

where  $L = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$

### 3.8.2 Logistic Regression Model

Pada sub bab ini terdapat fungsi yang menjelaskan tentang *simplified cost function* dan *gradient descent, decision boundary, hypothesis representation* dan *cost function*.

#### 3.8.2.1 Multiclass Classification

*Multiclass classification* akan mengklasifikasi data ketika data memiliki lebih dari dua kategori. Jika  $y = \{0,1\}$  akan diperluas menjadi  $y = \{0,1, \dots, n\}$ .

$y = \{0,1, \dots, n\}$  akan membagikan permasalahan ke dalam  $n+1$  (karena indeks dimulai dari 0).

Fungsi yang digunakan dalam menghitung *multiclass classification* adalah sebagai berikut:

$$y \in \{0,1, \dots, n\}$$

$$h_{\theta}^{(0)}(x) = P(y=0|x; \theta)$$

$$h_{\theta}^{(1)}(x) = P(y=1|x; \theta)$$

$$\dots$$

$$h_{\theta}^{(n)}(x) = P(y=n|x; \theta)$$

$$\text{prediction} = \max(h_{\theta}^{(0)}(x))$$

#### 3.8.2.2 Simplified Cost Function and Gradient Descent

Pada sub bab *simplified cost function* dan *gradient descent* ini terdapat fungsi untuk menghitung *simplified cost function* dan *gradient descent*.

Fungsi *simplified cost function* adalah sebagai berikut :

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

Sebuah vektorialisasi di implementasikan sebagai berikut :

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m} \cdot (-y^T \log(h) - (1-y)^T \log(1-h))$$

Bentuk umum *gradient descent* adalah sebagai berikut :

*Repeat*{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

Turunan dengan menggunakan kalkulus adalah sebagai berikut :

**Institut teknologi**

*Repeat*{

$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

Algoritma diatas identik dalam regresi *linear* yang secara bersamaan semua nilai diperbarui dalam *theta*.

Implementasi vektorialisasi adalah sebagai :

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

### 3.8.2.3 Decission Boundary

Untuk mendapatkan klasifikasi diskrit 0 atau 1 dapat mengartikan keluaran dari fungsi hipotesis sebagai berikut :

**2001**

$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$

$$h_\theta(x) < 0.5 \rightarrow y = 0$$

Fungsi *logistic*  $g$  bersifat masukan lebih besar dari atau sama dengan nol, keluaran lebih besar dari atau sama dengan 0.5:

$$g(z) \geq 0.5$$

*when z ≥ 0*

Fungsi yang perlu di ingat :

$$z = 0, e^0 = 1 \Rightarrow g(z) = 1/2$$

$$z \rightarrow \infty, e^{-\infty} \rightarrow 0 \Rightarrow g(z) = 1$$

$$z \rightarrow -\infty, e^{\infty} \rightarrow \infty \Rightarrow g(z) = 0$$

Jadi jika masukan adalah  $\theta^T X$ , itu artinya :

$$h_{\theta}(x) = g(\theta^T x) \geq 0.5$$

$$\text{when } \theta^T x \geq 0$$

Dari pernyataan diatas dapat disimpulkan bahwa :

$$\begin{aligned} \theta^T x \geq 0 &\Rightarrow y = 1 \\ \theta^T x < 0 &\Rightarrow y = 0 \end{aligned}$$

*Decision boundary* adalah garis yang memisahkan daerah  $y = 0$  dan  $y = 1$ .

### 3.8.2.4 Hypothesis Representation

Dalam *hypothesis representation* memasukkan  $\theta^T x$  ke dalam fungsi *logistic*.

Fungsi *sigmoid* atau yang disebut juga fungsi *logistic* adalah sebagai berikut :

$$h_{\theta}(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

### 3.8.2.5 Cost Function

Penggunaan *cost function* yang sama tidak dapat digunakan untuk regresi *linear* karena fungsi *logistic* akan menyebabkan keluaran menjadi tidak beraturan, menyebabkan banyak optima lokal. Dengan kata lain tidak akan menjadi *convex function*.

Fungsi *cost function* adalah sebagai berikut :

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_{\theta}(x), y) = -\log(h_{\theta}(x)) \quad \text{if } y = 1$$

$$\text{Cost}(h_{\theta}(x), y) = -\log(1 - h_{\theta}(x)) \quad \text{if } y = 0$$

### 3.9 Ringkasan

Berdasarkan analisis yang dilakukan, penulis merangkum hal-hal terkait *essay grading*, *features selection*, *linear regression*, *logistic regression* dalam butir-butir sebagai berikut :

- Model *linear* dan *logistic regression* bekerja cukup baik dalam *essay grading*
- Dataset dari *Kaggle Competition* sudah mengalami *preprocessing* yang cukup baik sehingga akan membantu pembentukan model prediktif.
- *Features scaling* dapat mengoptimalkan pembentukan model
- Kekurangan *forward* dan *backward* dalam *features selection* dapat diatasi dengan menggunakan *wrapper method*
- Model *evaluation* yang digunakan adalah *Root Mean Squared Error* (RMSE) dan *Quadratic Weighted Kappa*



## Bab IV Desain dan Implementasi Eksperimen

Pada Bab desain eksperimen ini terdapat penjelasan mengenai deskripsi umum, lingkungan eksperimen, tahapan eksperimen, dan deskripsi program.

### 4.1 Deskripsi Umum

Secara umum, eksperimen yang akan dilakukan terbagi atas 2 eksperimen, yakni:

- Eksperimen terhadap metode *wrapper* pada proses *features selection*
- Eksperimen terhadap penambahan parameter *n-grams*

Eksperimen terhadap metode *wrapper* pada proses *features selection* bertujuan untuk memilih *features* yang memiliki kontribusi maksimal terhadap akurasi prediksi skor esai, sedangkan eksperimen terhadap penambahan parameter *n-grams* bertujuan untuk mengetahui apakah penambahan parameter tersebut dapat meningkatkan akurasi prediksi skor esai. Jadi eksperimen yang melibatkan *n-grams* hanya dapat dilakukan apabila eksperimen yang melibatkan metode *wrapper* untuk *features selection* yang telah selesai dilakukan.

### 4.2 Lingkungan Eksperimen

Berikut merupakan penjelasan mengenai lingkungan operasional yang digunakan untuk melakukan eksperimen dan pengembangan program kecil.

#### 1. Hardware

Spesifikasi *hardware* yang digunakan adalah:

- Processor : Intel Core i-5
- Memory : 10.00 GB RAM
- Hard disk : 500 GB

## 2. Software

Spesifikasi *software* yang digunakan adalah:

1. Sistem Operasi : Windows 10
2. *Editor* : PyCharm 2016.3, Spyder 3.1.4
3. *Package Manager* : Anaconda 2.4.0
4. *Documentation* : Microsoft Office 2013
5. Bahasa Pemograman : Python 2.7

Kedua eksperimen yang telah disebutkan pada sub bab 4.2 akan dilakukan pada *environtment* yang sama.

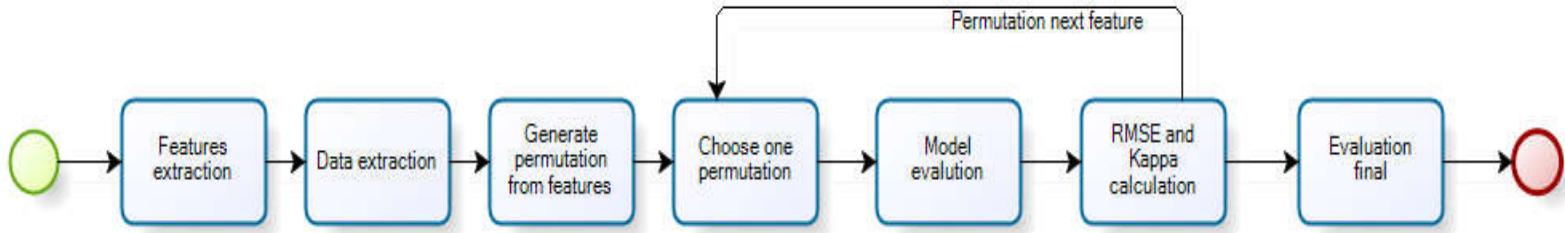
### 4.3 Tahapan Eksperimen

Eksperimen untuk Tugas Akhir ini dilakukan secara bertahap. Tahapan-tahapan dalam melakukan eksperimen ini dirancang agar proses pencapaian tujuan Tugas Akhir ini dapat berjalan dengan terarah. Berikut merupakan tahapan-tahapan eksperimen yang akan dilakukan untuk kedua eksperimen yang disebutkan pada sub bab 4.2.

#### 4.3.1 Tahapan Eksperimen terhadap metode Wrapper pada proses features selection

Pada sub bab ini terdapat penjelasan dari tahapan eksperimen terhadap metode *wrapper* pada proses *features selection* dengan syarat data telah di *preprocessing*, dapat dilihat pada gambar 8.

Eksperimen Terhadap Metode Wrapper pada Proses Features Selection



Gambar 8. Tahapan eksperimen terhadap metode *wrapper* pada proses *features selection*

MARTUHAN-MARROHA-MARBISUK

2001

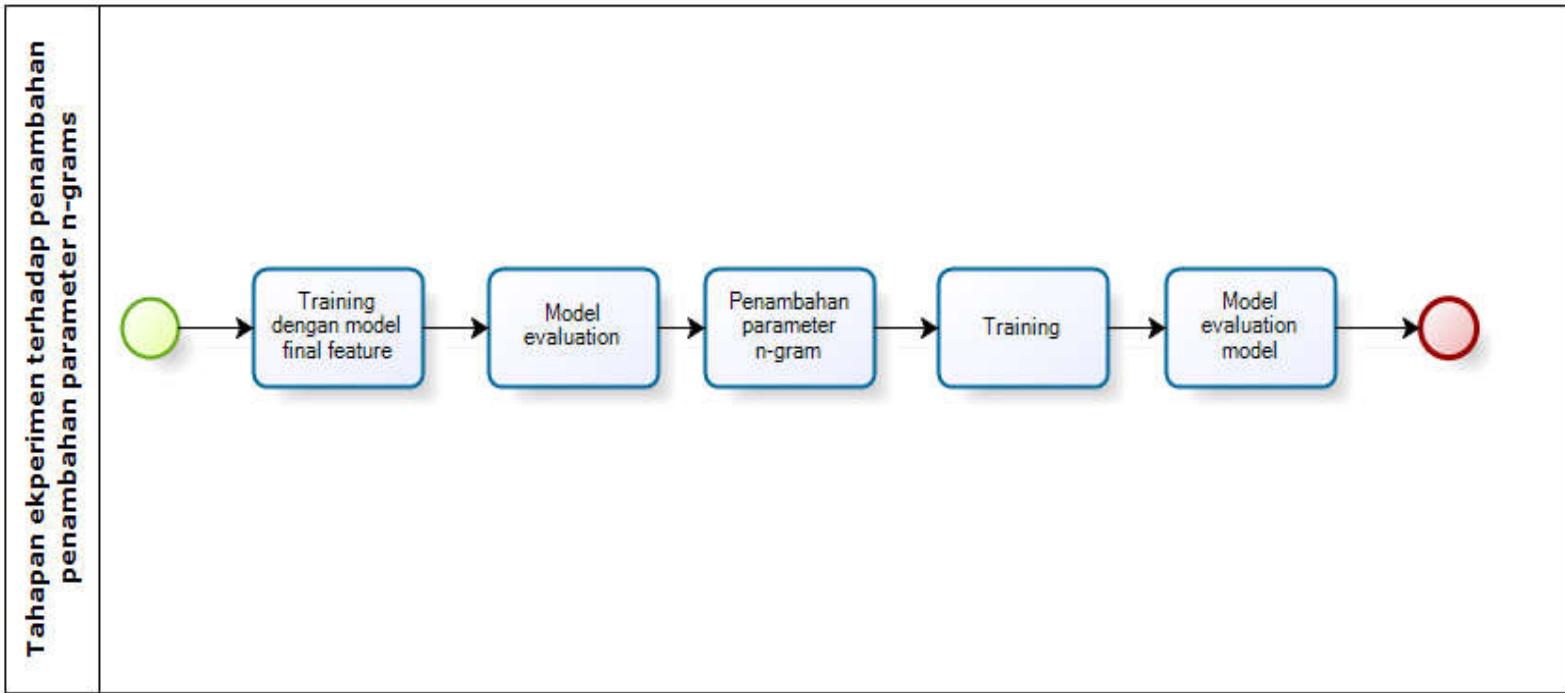
Keterangan :

- 1.1 *Features Extraction* : dilakukan untuk mendapatkan *features* yang akan digunakan untuk memprediksi skor esai. *Features* ini di ekstrak dengan menggunakan bantuan *packages* yang telah dijelaskan pada sub bab 3.4. Setiap esai pada dataset memiliki *features* tersendiri.
- 1.2 *Data Extraction Value*: merupakan tahapan dimana setiap data diberi label-label yang merupakan nilai dari *feature* yang diekstrak pada tahap sebelumnya.
- 1.3 Pemilihan data untuk evaluasi *features*: merupakan proses pemilihan subset dari data keseluruhan yang dilakukan dengan pertimbangan bahwa data yang dipilih harus terdistribusi normal.
- 1.4 Pembagian Data: dilakukan dengan tujuan untuk membagi data menjadi 2 bagian, yakni *training set* untuk kepentingan proses *training model* dan *test set* untuk keperluan evalusi model. Adapun komposisi pembagian data adalah 70% dan 30% *test set*.
- 1.5 *Generate Permutasi* dari *feature* : merupakan proses untuk meng-*generate* semua kemungkinan permutasi dari kombinasi *feature* yang tersedia.
- 1.6 Pilih salah satu permutasi dari *feature* dan lakukan proses *training*: tahap ini dilakukan secara *iteratif* untuk semua permutasi dari *feature* .
- 1.7 *Model Evaluation*: sama dengan tahap 1.6, tahap ini dilakukan secara *iteratif* untuk semua permutasi dari *feature* dengan tujuan mengukur tingkat akurasi dari model yang di *train*.
- 1.8 Perhitungan *RMSE*
- 1.9 Evaluasi *Final*

#### **4.3.2 Tahapan Eksperimen terhadap penambahan parameter n-grams**

Pada tahapan ini dilakukan proses yang hampir sama dengan proses eksperimen terhadap *features-selection* pembedanya adalah pada eksperimen ini model sudah tersedia.

Tahapan ini dilakukan dengan menambahkan *n-grams feature* pada model *final* dimana *n-grams feature* yang digunakan hanyalah *uni-grams*, *bi-grams* dan *tri-grams*. Skema tahapannya dapat dilihat pada gambar 9.



Gambar 9. Tahapan eksperimen terhadap penambahan parameter *n-grams*

MARTUHAN-MARROHA-MARBISUK

2001

Keterangan :

- 2.1 *Generate* Permutasi dari *feature* : merupakan proses untuk meng-*generate* semua kemungkinan permutasi dari kombinasi *feature* yang tersedia.
- 2.2 Pilih salah satu permutasi dari *feature* dan lakukan proses *training* : tahap ini dilakukan secara *iteratif* untuk semua permutasi dari *feature*.
- 2.3 *Model Evaluation* : sama dengan tahap 1.6, tahap ini dilakukan secara *iteratif* untuk semua permutasi dari *feature* dengan tujuan mengukur tingkat akurasi dari model yang di *train*.

#### 4.4 Deskripsi Program

Pada sub bab deskripsi program menjelaskan tentang *preprocessing*, *feature extraction*, *feature selection*, *training model*, dan *features scaling*.

##### 4.4.1 Preprocessing

*Preprocessing* merupakan tahapan awal dalam mengolah esai sebelum memasuki proses tahapan *feature extraction*. *Preprocessing* esai dilakukan untuk tujuan penyeragaman dan kemudahan pembacaan. *Preprocessing* terdiri dari beberapa tahapan. Adapun tahapan *preprocessing* yang dilakukan yaitu: *case folding*, *tokenizing* dan *stopword removal* yang telah dijelaskan sebelumnya pada sub bab 3.3. dan implementasi dari pada *preprocessing* adalah sebagai berikut.

###### 1. Case Folding

Fungsi dari *case folding* terdapat pada lampiran program dengan kode fungsi PFX03. Pada fungsi tersebut parameter yang diterima berupa esai *string* dan melakukan perubahan ke huruf kecil dengan menggunakan fungsi dari bahasa pemrograman *python*.

###### 2. Tokenization

Fungsi dari *tokenization* terdapat pada lampiran program dengan kode fungsi PFX04. Pada fungsi tersebut parameter yang diterima berupa esai *string* dan melakukan tokenisasi dengan menggunakan fungsi *findall* dari *library regular expression operations*. *Findall* adalah sebuah fungsi yang melakukan *scanning* data dari kiri kenan dan menampung data kedalam sebuah *list string*. Parameter '/w' dalam fungsi *findall* melakukan penghapusan pada karakter yang tidak alfanumerik seperti tanda baca.

### 3. Stopword Removal

Fungsi dari *stopword removal* terdapat pada lampiran program dengan kode fungsi PFX02. Pada fungsi tersebut parameter yang diterima berupa *list string* yang telah di tokenisasi. Pada fungsi ini dilakukan penghapusan sebuah *string* berupa kata dari *stopword* dengan menggunakan *library stopword*. *Library* ini menyediakan fungsi *clean* dengan dua parameter yaitu *list* dari *string* yang ingin dihapus *stopword* nya dan kode bahasa yang digunakan seperti kode ‘en’ yang berarti bahasa inggris.

#### 4.4.2 Feature Extraction

*Feature extraction* atau ekstraksi *feature* merupakan suatu pengambilan ciri/*feature* dari suatu bentuk yang menghasilkan nilai yang berguna untuk proses selanjutnya. *Feature extraction* dilakukan dengan cara mengkalkulasikan setiap esai dengan 16 *feature* yang dikelompok kan pada sub bab 3.4. Perhitungan dari *feature extraction* dilakukan dengan pengansumsian dikarenakan tidak ada penjelasan terkait hal tersebut di dalam jurnal sebelumnya. Berikut implementasi dari pada *feature extraction*.

##### 1. Word Count

Fungsi dari *word count* terdapat pada lampiran program dengan kode fungsi PFX 09, fungsi ini bertujuan menghitung jumlah kata yang unik pada esai. Pada fungsi tersebut parameter yang diterima berupa *list string* yang telah di tokenisasi. Di dalam fungsi tersebut terdapat fungsi dari *library collections* yaitu *counter* untuk melakukan *hashable* pada setiap *element string* pada *list* dan melakukan perhitungan jumlah elemen untuk mendapatkan jumlah kata yang unik pada esai.

##### 2. Long Word Count

Fungsi dari *long word count* terdapat pada lampiran program dengan kode fungsi PFX 06, fungsi ini bertujuan menghitung jumlah kata yang panjang kata nya lebih besar atau sama dengan dari panjang rata-rata kata yang unik dalam sebuah esai. Pada fungsi tersebut parameter yang diterima berupa *list string* yang telah di tokenisasi. Di dalam fungsi tersebut terdapat fungsi *set* yang bertujuan untuk mendapatkan kata yang unik selanjutnya melakukan perhitungan panjang rata-rata dan menghitung jumlah kata yang lebih besar darin panjang rata-rata.

### 3. Part Of Speech Count

Fungsi dari *Part of Speech Count* terdapat pada lampiran program dengan kode fungsi PFX 16, fungsi ini bertujuan menghitung jumlah *noun*, *verb*, *adverb*, dan *adjective* pada esai. Didalam fungsi ini terdapat fungsi dari *library NLTK pos tagger* yang berguna memberikan label pada tiap kata. Pada fungsi tersebut parameter yang diterima berupa *list string* yang telah di tokenisasi, memberikan *tagger* pada tiap kata dan melakukan perhitungan, pada *tagger* kata yang diawali dengan kata kunci NN adalah *noun*, VB adalah *verb*, RB adalah *adverb* dan JJ adalah *adjective*.

### 4. Comma Count

Fungsi dari *comma count* terdapat pada lampiran program dengan kode fungsi PFX10. Pada fungsi tersebut parameter yang diterima berupa esai *string* dan melakukan perhitungan jumlah *comma* pada esai dengan menggunakan fungsi dari bahasa pemrograman *python*.

### 5. Punctuation Count

Fungsi dari *punctuation count* terdapat pada lampiran program dengan kode fungsi PFX15. Fungsi ini bertujuan untuk menghitung jumlah tanda baca pada esai. Pada fungsi tersebut parameter yang diterima berupa esai *string* dan melakukan perhitungan jumlah tanda baca dengan melakukan pengecekan terhadap *list punctuation* yang disediakan oleh bahasa pemrograman *python*.

### 6. Sentence Count

Fungsi dari *sentence count* terdapat pada lampiran program dengan kode fungsi PFX14. Fungsi ini bertujuan untuk menghitung jumlah kalimat pada esai. Pada fungsi tersebut parameter yang diterima berupa esai *string* dan melakukan perhitungan dengan pengecekan pada tanda titik, tanda seru dan tanda tanya.

## 7. *Lexical Diversity Count*

Fungsi dari *lexical diversity* terdapat pada lampiran program dengan kode fungsi PFX08. Fungsi ini bertujuan untuk menghitung jumlah keberagaman bahasa pada esai. Pada fungsi tersebut parameter yang diterima berupa *list string* dari esai yang telah di tokenisasi dan melakukan perhitungan dengan jumlah kata yang *distinct* dibagi dengan banyak seluruh kata pada esai.

## 8. *Quotation Mark Count*

Fungsi dari *Quotation mark* terdapat pada lampiran program dengan kode fungsi PFX11. Fungsi ini bertujuan untuk menghitung jumlah kalimat kutipan atau penekanan pada esai. Pada fungsi tersebut parameter yang diterima berupa *string* dan melakukan perhitungan dengan fungsi dari *library regular expression operations*. *Findall* adalah sebuah fungsi yang melakukan *scanning* data dari kiri ke kanan dan menampung data kedalam sebuah *list string*.

## 9. *Word Length Count*

Fungsi dari *word length* terdapat pada lampiran program dengan kode fungsi PFX07. Fungsi ini bertujuan untuk menghitung jumlah panjang kata yang tidak duplikat dibagi dengan jumlah kata pada esai. Pada fungsi tersebut parameter yang diterima berupa *list string* dan melakukan perhitungan.

## 10. *Bracket Count*

Fungsi dari *bracket count* terdapat pada lampiran program dengan kode fungsi PFX12. Fungsi ini bertujuan untuk menghitung jumlah kalimat yang berada didalam *bracket* atau tanda kurung pada esai. Pada fungsi tersebut parameter yang diterima berupa *string* dan melakukan perhitungan.

## 11. *Spell Error Count*

Fungsi dari *Spell error count* terdapat pada lampiran program dengan kode fungsi PFX17. Fungsi ini bertujuan untuk menghitung jumlah kata yang salah eja pada esai. Pada fungsi tersebut parameter yang diterima berupa *list string* dan melakukan pengecekan, jika kata tersebut dapat tidak sama dengan *spelling replacer* berarti kata tersebut merupakan kata yang salah dalam pengejaan bahasa inggris.

## 12. *Exclamation Marks Count*

Fungsi dari *exclamation marks count* terdapat pada lampiran program dengan kode fungsi PFX13. Fungsi ini bertujuan untuk menghitung jumlah kalimat yang memiliki tanda seru. Pada fungsi tersebut parameter yang diterima berupa *string* dan melakukan perhitungan dengan *string count*.

### 13. Foreign Word Count

Fungsi dari *foreign word count* terdapat pada lampiran program dengan kode fungsi PFX18. Fungsi ini bertujuan untuk menghitung jumlah kata yang bukan bahasa Inggris. Pada fungsi tersebut parameter yang diterima berupa *list string* dan melakukan perhitungan dengan melakukan pengecekan dengan *library enchant* apakah kata yang di cek termasuk dalam kata bahasa Inggris atau tidak.

### 14. N-Grams Count

Fungsi dari *n-grams count* terdapat pada lampiran program dengan kode fungsi PFX19. Fungsi ini bertujuan untuk menghitung jumlah *bi-grams* dan *tri-grams* yang terdapat pada esai. Pada fungsi tersebut parameter yang diterima berupa *list string* dan melakukan perhitungan dengan melakukan pengecekan dengan *library NLTK*.

## 4.4.3 Features Selection

Pada sub bab *features selection* terdapat penjelasan dari *forward features selection* dan *wrapper features selection*.

### 4.4.3.1 Forward Features Selection

Fungsi dari *forward feature selection* terdapat pada lampiran program dengan kode fungsi FSEL 01 dan FSEL 02. Kode fungsi FSEL 01 adalah fungsi untuk melakukan pemilihan *feature* dengan menggunakan nilai kappa dan kode fungsi FSEL 02 adalah fungsi untuk melakukan pemilihan *feature* dengan menggunakan nilai RMSE. Dalam fungsi tersebut terdapat 3 parameter yaitu x yang merupakan matriks *features* dan y merupakan *vector target*, dan model adalah pemodelan yang dipilih regresi *linear* atau regresi *logistic*. Fungsi ini bertujuan untuk melakukan proses *forward feature selection*.

### 4.4.3.2 Wrapper Features Selection

Tahapan ini diimplementasikan dengan beberapa fungsi yakni *generateSubset* yakni fungsi yang bertugas untuk menghasilkan keseluruhan kombinasi subset dari *feature*,

untuk detail implementasi fungsi dapat dilihat pada lampiran program dengan kode fungsi FSEL 03, dan fungsi *wrapperBruteForceFeaturesSelection* yang bertugas untuk melakukan *wrapper features selection*.

#### 4.4.4 Training Model

Pada sub bab *training model* terdapat penjelasan dari *single feature linear regression* dan *train and cross validate*.

##### 4.4.4.1 Single Feature Linear Regression

Fungsi ini terdiri dari dua fungsi yakni *train AndMeasureLinearRegressionSingleFeature* dan *train AndMeasureLogisticRegressionSingleFeature*, implementasi fungsi ini dapat dilihat pada lampiran program dengan kode fungsi TCVME 03 dan TCVME 04. Fungsi ini menerima X yang merupakan matriks *feature* dan Y sebagai *vector target*, lalu mengkalkulasi hasil akurasi skor *kappa* dan *rmse* untuk setiap *feature* secara independen.

##### 4.4.4.2 Train and Cross Validate

Fungsi ini terdiri dari dua fungsi yakni *trainAndMeasureLinearRegressionCrossValidate* dan *trainAndMeasureLogisticRegressionCrossValidate*, implementasi fungsi ini dapat dilihat pada lampiran program dengan kode fungsi TCVME 01 dan TCVME 02.

Fungsi ini menerima X yang merupakan matriks *feature* dan Y sebagai *vector target*, lalu mengkalkulasi hasil akurasi skor *kappa* dan *rmse* untuk keseluruhan matriks X dengan melakukan *10-fold-cross-validation*.

## Bab V Hasil dan Pembahasan

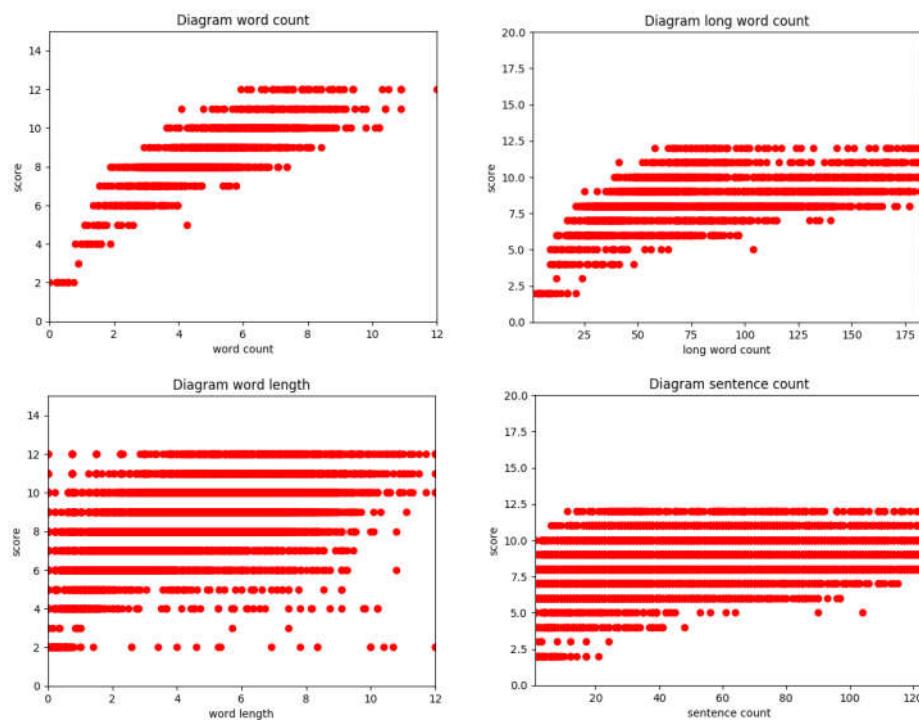
Pada Bab hasil dan pembahasan distribusi data terhadap skor esai, hasil akurasi model terhadap *single feature*, hasil dari *forward features selection*, penambahan parameter *n-grams* terhadap *forward method*, *wrapper features selection*, dan penambahan parameter *n-grams* terhadap *wrapper method*.

### 5.1 Distribusi Data

Pada sub bab distribusi data ini membahas tentang distribusi data antara *feature* dalam esai terhadap skor esai. Distribusi data dikelompokkan berdasarkan *word and sentence*, *part of speech tagger*, *structure and orthograpgy*, *foreign word and lexical diversity*, dan *misspelt error*.

#### 5.1.1 Word and Sentence

Berikut merupakan diagram *word count*, *long word count*, *average word length*, dan *sentence count*:

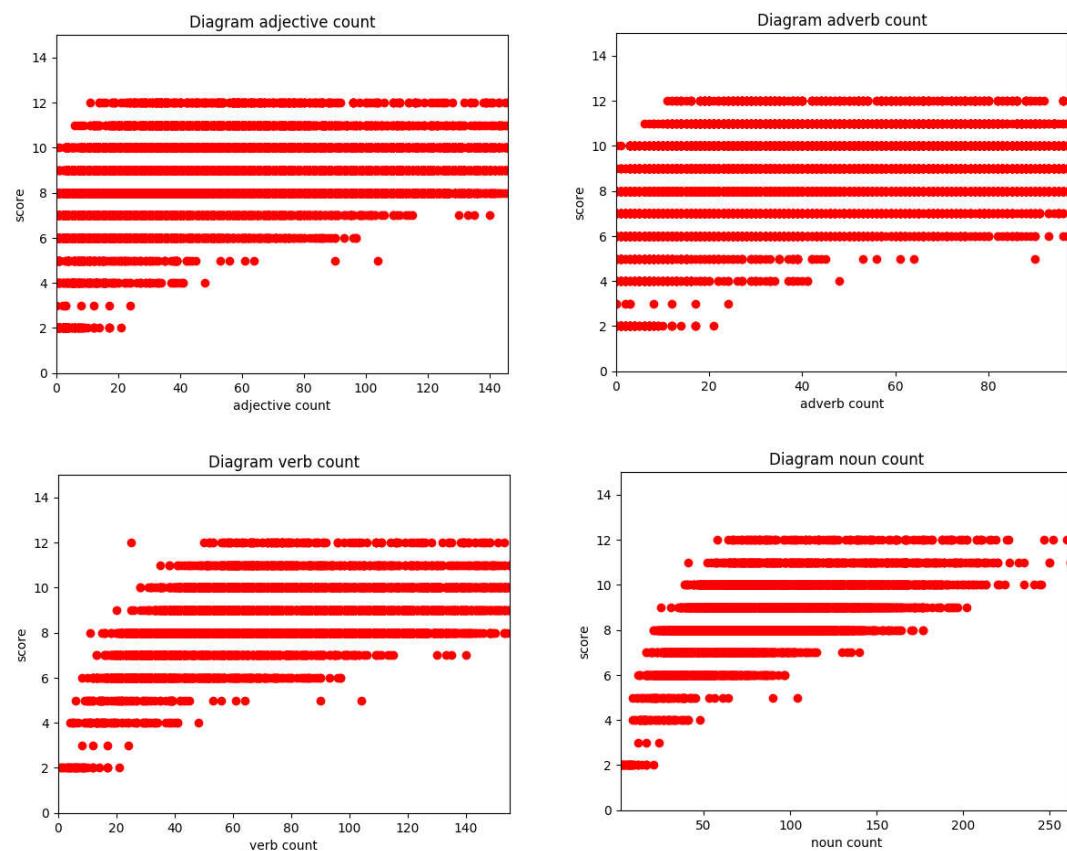


Gambar 10. Numerical features

Pada gambar 10 diatas dapat dilihat bahwa distribusi data terhadap skor esai memiliki nilai keterikatan yang cukup tinggi, terutama *word count* dan *long word count* terhadap skor esai menunjukkan skala berbanding lurus, meskipun terdapat beberapa data yang terdistribusi menyimpang terhadap skor esai. Dari persebaran data diatas dapat disimpulkan bahwa *features* berbasis kata dan kalimat dapat memberi bobot prediktif yang cukup signifikan terhadap prediksi skor esai.

### 5.1.2 Part of Speech Tagger

Berikut merupakan diagram *adjective count*, *adverb count*, *noun count* dan *verb count* :



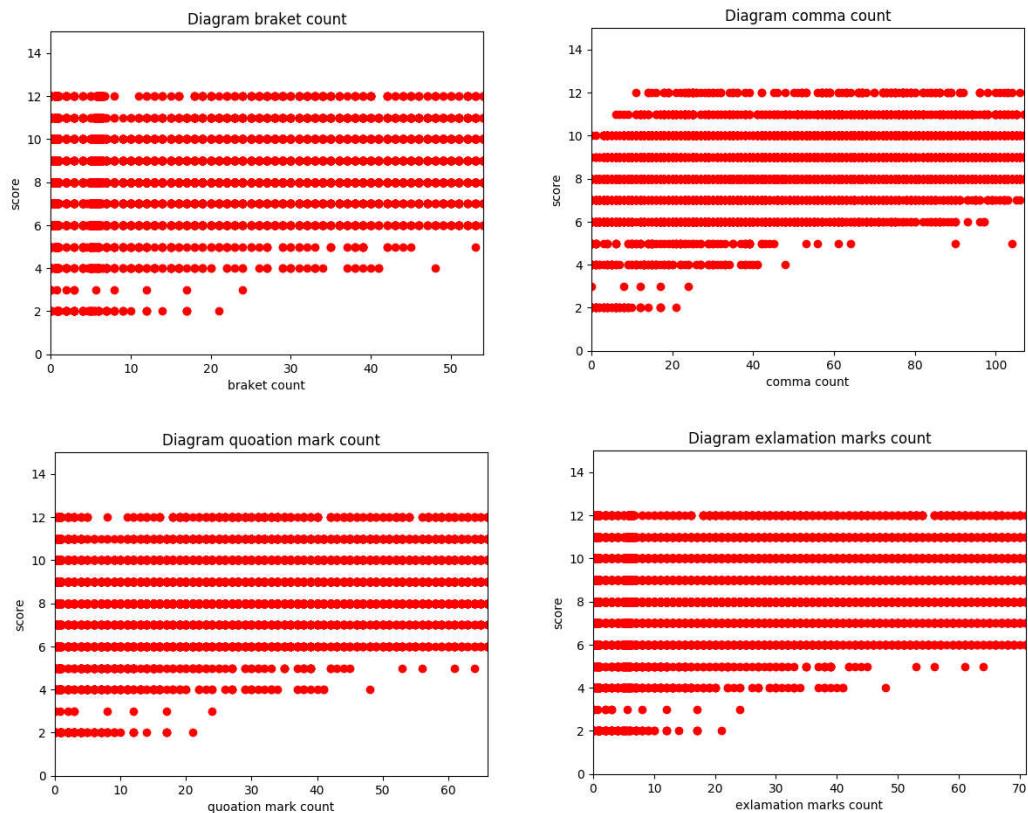
Gambar 11. *Part of speech tagger*

Pada gambar 11 diatas dapat dilihat bahwa distribusi data terhadap skor esai memiliki nilai keterikatan yang cukup tinggi untuk *noun count* dan *verb count* akan tetapi untuk *feature adjective count* dan *adverb count* data terdistribusi tidak sebaik *noun* dan *verb count*, dengan pertimbangan bahwa kedua *features* tersebut pada dasarnya memiliki

koralasi yang cukup terikat satu sama lain, yakni dalam *term part-of-speech-tagger*, *feature* tersebut tentunya memiliki bobot predikif yang cukup untuk prediksi skor esai.

### 5.1.3 Structure and Orthography

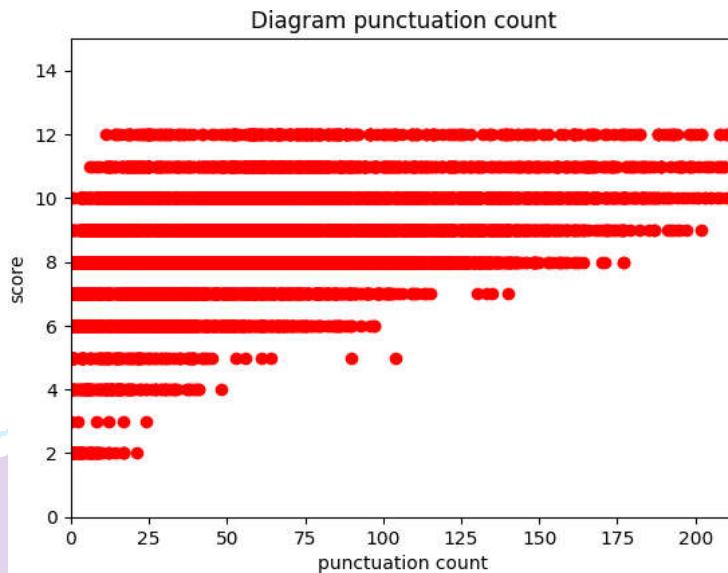
Berikut merupakan diagram *bracket count*, *comma count*, *quotation mark count* dan *exclamation mark*:



Gambar 12. *Structure and orthography*

Pada gambar 12 diatas dapat dilihat bahwa distribusi data untuk keempat *features* tanda baca, tidak begitu baik. Distribusi tidak memberikan *boundary* antara skor 1, 2, 3 dst pada esai terhadap *count* dari *feature* tersebut.

Bahkan setelah dilakukan penggabungan untuk seluruh tanda baca seperti pada gambar berikut :

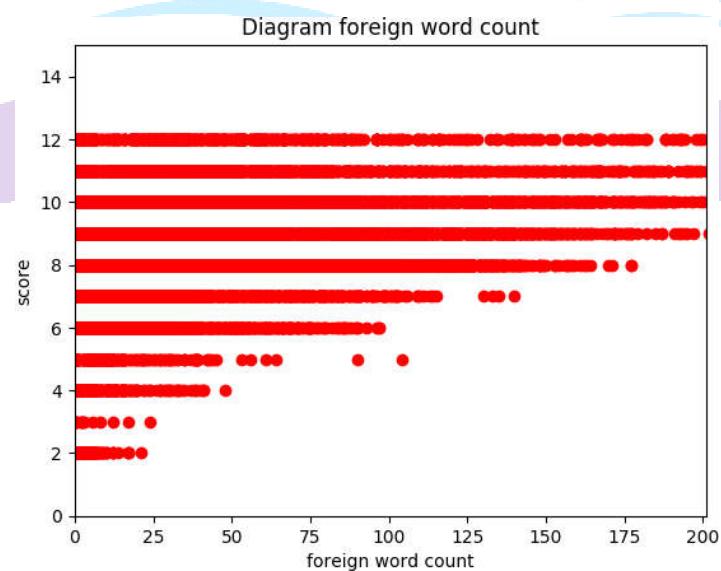


Gambar 13. *Overall punctuation count*

Dari gambar 13 dapat dilihat bahwa distribusi data sudah memiliki keterikatan terhadap skor esai, hal ini disebabkan struktur suatu kalimat tidak terlalu tergantung pada satu tanda baca, akan tetapi beberapa gabungan tanda baca yang berkorelasi dengan diatur berbasis kata dan kalimat.

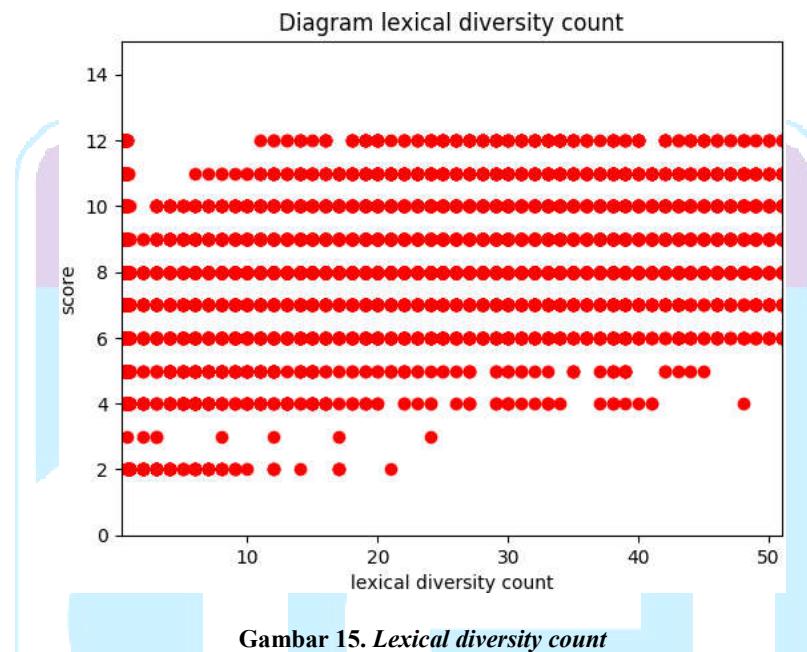
#### 5.1.4 Foreign Word and Lexical Diversity

Berikut merupakan diagram *foreign word count* dan *lexical diversity* :



Gambar 14. *Foreign word count*

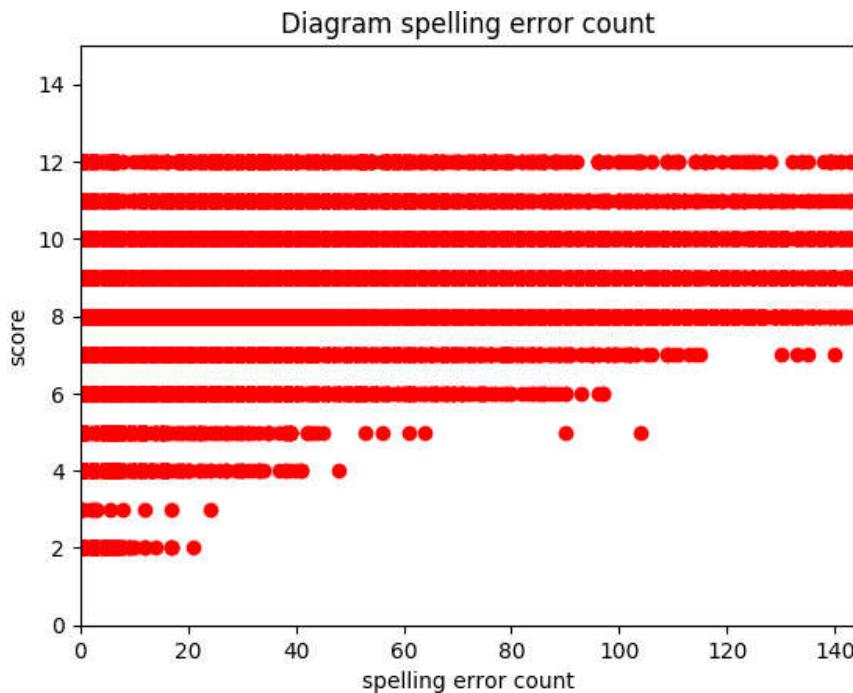
Pada gambar 14 diatas dapat dilihat bahwa distribusi data terhadap skor esai memiliki nilai keterikatan yang cukup tinggi terutama dalam menghitung jumlah kata yang bukan bahasa inggris. Dari persebaran data diatas dapat disimpulkan bahwa *features* berbasis kata dan kalimat dapat memberi bobot prediktif yang cukup signifikan terhadap prediksi skor esai.



Pada gambar 15 terdapat data distribusi data terhadap *score essay* memiliki nilai keterikatan yang cukup tinggi, untuk *feature noun, verb, adverb* terutama *word count* dan *long word count* terhadap skor esai menunjukkan skala berbanding lurus, meskipun terdapat beberapa data yang terdistribusi menyimpang terhadap skor esai. Dari persebaran data diatas, dapat disimpulkan bahwa *features* berbasis kata dan kalimat dapat memberi bobot prediktif yang cukup signifikan terhadap prediksi skor esai.

### 5.1.5 Misspelt Error

Berikut merupakan diagram *orthography misspelt errors* :



Gambar 16. *Orthography misspelt errors*

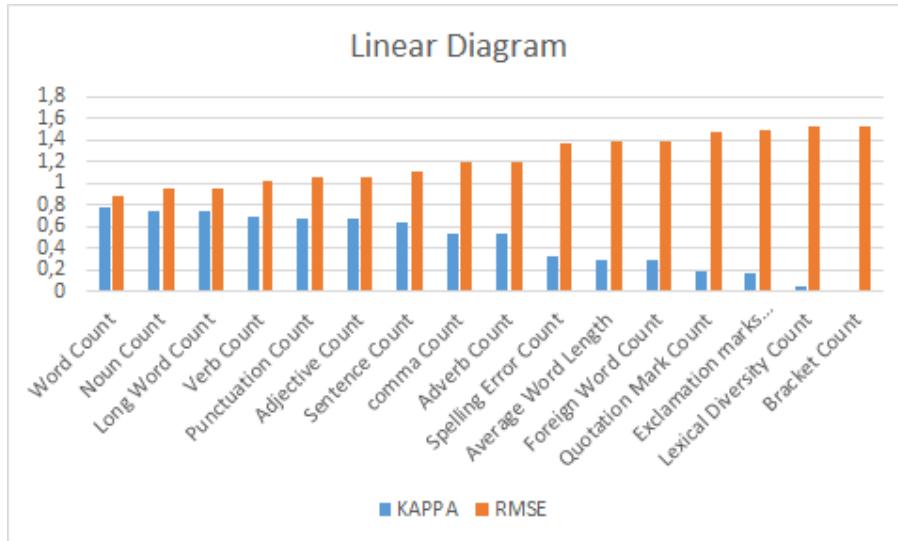
Pada gambar 16 terdapat data distribusi data terhadap *score essay* memiliki nilai keterikatan yang cukup tinggi, untuk *feature noun, adverb*, terutama *word count* dan *long word count* terhadap skor esai menunjukkan skala berbanding lurus, meskipun terdapat beberapa data yang terdistribusi menyimpang terhadap skor esai. Dari persebaran data diatas, dapat disimpulkan bahwa *features* berbasis kata dan kalimat dapat memberi bobot prediktif yang cukup signifikan terhadap prediksi skor esai.

## 5.2 Single Feature Model Evaluation

Pada penelitian ini dilakukan *single feature model evaluation* terhadap kedua model yakni *linear* dan *logistic*. Adapun pengukuran dari model tersebut adalah sebagai berikut.

### 5.2.1 Linier Regression Model

Pada tahap ini dilakukan pengukuran dengan kappa dan RMSE.

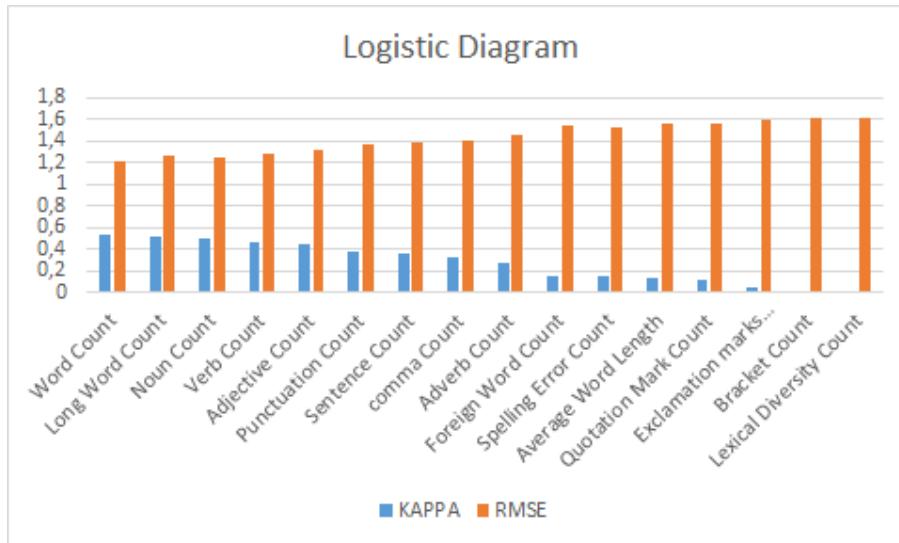


Gambar 17. Linear regression model

Pada gambar 17 dapat dilihat bahwa pengukuran yang dilakukan dengan kappa masing-masing *feature* terlebih dahulu melakukan pengurutan dari nilai tingkat tertinggi ke nilai tingkat terendah sedangkan RMSE merupakan nilai kuadrat kesalahan. Pada tahap pengukuran RMSE terlebih dahulu melakukan pengurutan dari nilai RMSE dari nilai tingkat terendah ke nilai tingkat tertinggi.

### 5.2.2 Logistic Regression Model

Pada tahap ini proses yang dilakukan sama hal nya dengan proses perhitungan pada kappa score regresi *linear* pada sub bab sebelumnya yaitu 5.2.1 perbedaannya adalah pada tahap ini dilakukan dengan regresi *logistic*.



Gambar 18. Logistic regression model

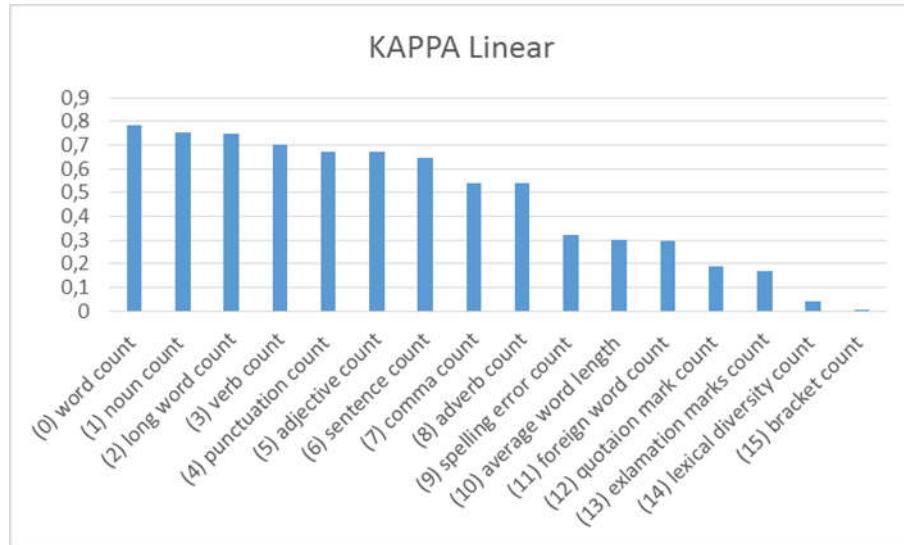
Pada gambar 18 dapat dilihat bahwa pengukuran dengan menggunakan kappa bahwa nilai dengan tingkat akurasi tertinggi adlaha *word count*, sedangkan dari RMSE nilai dengan tingkat akurasi tertinggi adalah *lexical diversity count*.

### 5.3 Forward Feature Selection

Pada penelitian ini dilakukan *forward feature selection* terhadap kedua model yakni *linear* dan *logistic*, masing-masing pada model dievaluasi dengan menggunakan kappa dan RMSE sebagai *error metric*. Adapun pengukuran dari model tersebut adalah sebagai berikut.

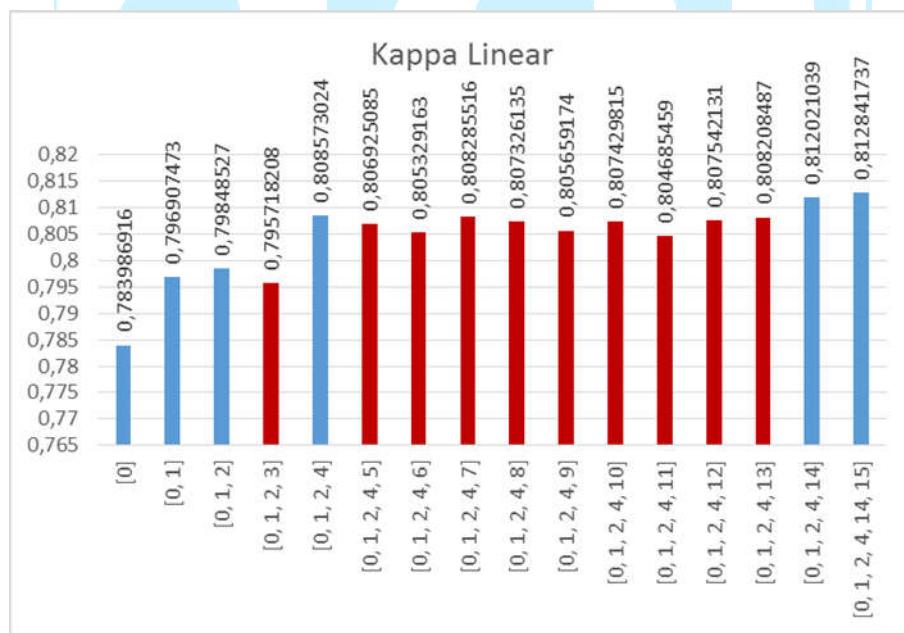
#### 5.3.1 Linear Regression Model Measured by Kappa Score

Pada tahap ini dilakukan pengukuran dengan menggunakan kappa, pengukuran dilakukan dengan masing-masing *feature* terlebih dahulu dan melakukan pengurutan dari nilai tingkat akurasi yang tertinggi sampai ke terendah. Dapat dilihat pada gambar 19, bahwa *feature* *word count* memiliki nilai dengan tingkat akurasi tertinggi disusul oleh *noun count* sampai *bracket count* dan pada masing masing *feature* diidentifikasi dengan *index feature* yang berada di dalam kurung pada nama *feature*.



Gambar 19. Pengurutan dari nilai tingkat akurasi yang tertinggi sampai ke terendah

Setelah dilakukan pengurutan dilakukan *forward feature selection*, pada *forward selection* setiap penambahan *feature* yang memiliki nilai tingkat akurasi yang lebih rendah maka *feature* tersebut tidak akan dimasukan. Dapat dilihat pada gambar 20. bagaimana proses *forward selection* dilakukan.



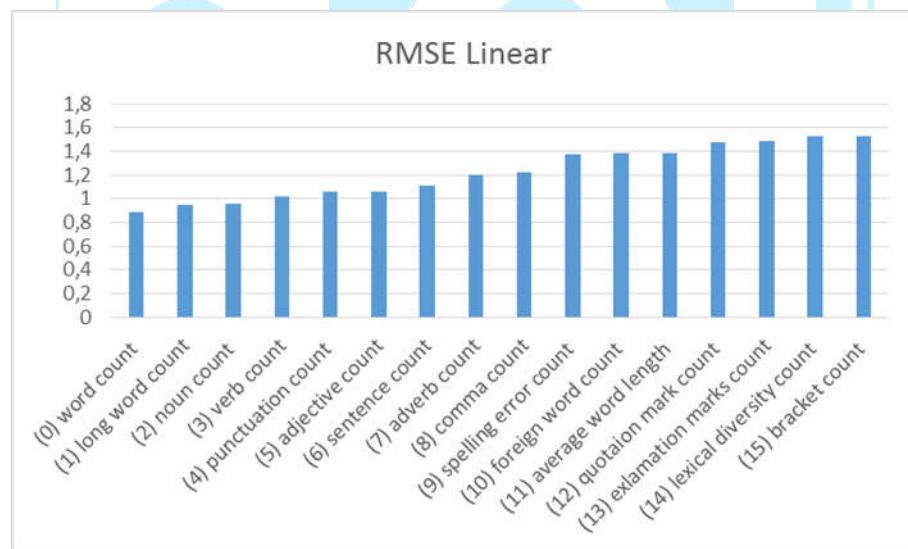
Gambar 20. Proses *forward selection*

Pada gambar 20 nama yang berada pada absis x adalah *index* dari masing masing *features* yang dijadikan sebagai variabel terhadap penilaian skor esai. Grafik yang berwarna merah menandakan bahwa penambahan suatu *feature* memiliki nilai tingkat akurasi yang

lebih rendah pada sebelumnya, sehingga *feature* tersebut tidak dimasukan kedalam perhitungan. Contohnya pada grafik yang ke empat dari diagram diatas adalah [0,1,2,3], penambahan *feature* dengan *index feature* nomor 3 menandakan bahwa *feature* tersebut memiliki tingkat akurasi yang lebih rendah dengan grafik yang sebelumnya yaitu [0,1,2] sehingga *feature* dengan *index* nomor 3 tidak dimasukan kedalam perhitungan. Dan *feature* akhir dalam proses ini adalah [0,1,2,4,14,15] yaitu *word count*, *noun count*, *long word count*, *verb count*, *punctuation count*, *lexical diversity count*, dan *bracket count*.

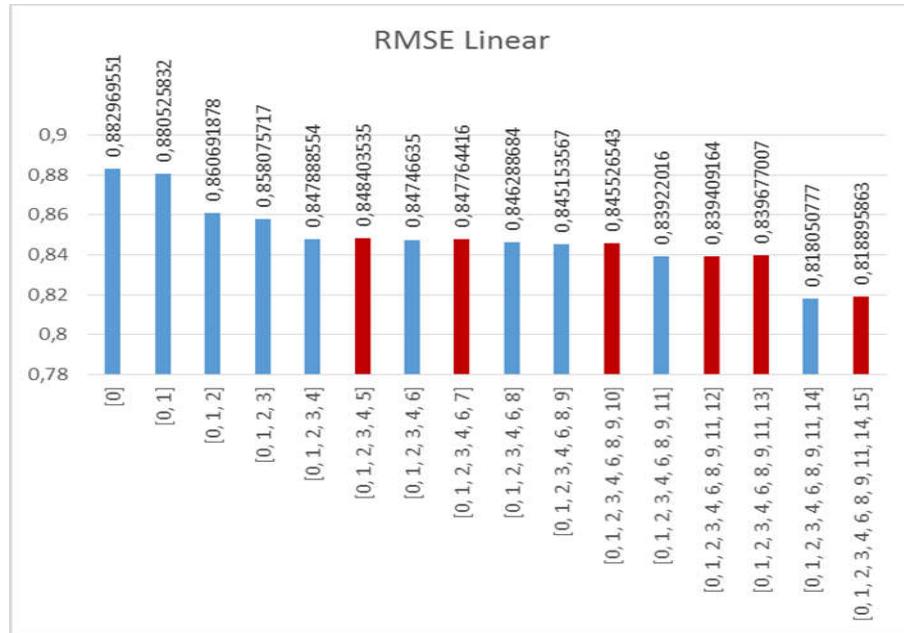
### 5.3.2 Linear Regression Model Measured by RMSE Score

Pada tahap ini dilakukan pengukuran dengan menggunakan RMSE. RMSE sendiri merupakan nilai kuadrat kesalahan. Nilai RMSE yang rendah menunjukkan bahwa variasi nilai yang dihasilkan oleh suatu model perkiraan mendekati variasi nilai observasinya. Dan pada tahap ini dilakukan pengukuran model dengan masing masing *feature* terlebih dahulu dan dilakukan pengurutan dari nilai RMSE yang rendah sampai ke tinggi. Dapat dilihat pada gambar 21 bagaimana perhitungan dilakukan, dan nomor pada nama *feature* adalah *index* dari nama *feature* tersebut.



Gambar 21. Pengurutan dari nilai RMSE yang rendah sampai ke tinggi

dapat dilihat pada gambar 21 bahwa *feature* *word count* memiliki nilai RMSE yang terendah dibandingkan dengan *feature* *bracket count* yang memiliki nilai RMSE tertinggi. Dan selanjutnya dilakukan *forward selection* dalam pemilihan *feature*, *feature* yang ditambahkan yang menghasilkan nilai yang lebih tinggi tidak akan digunakan. Dapat dilihat pada gambar 22 bagaimana *forward feature* RMSE dilakukan.



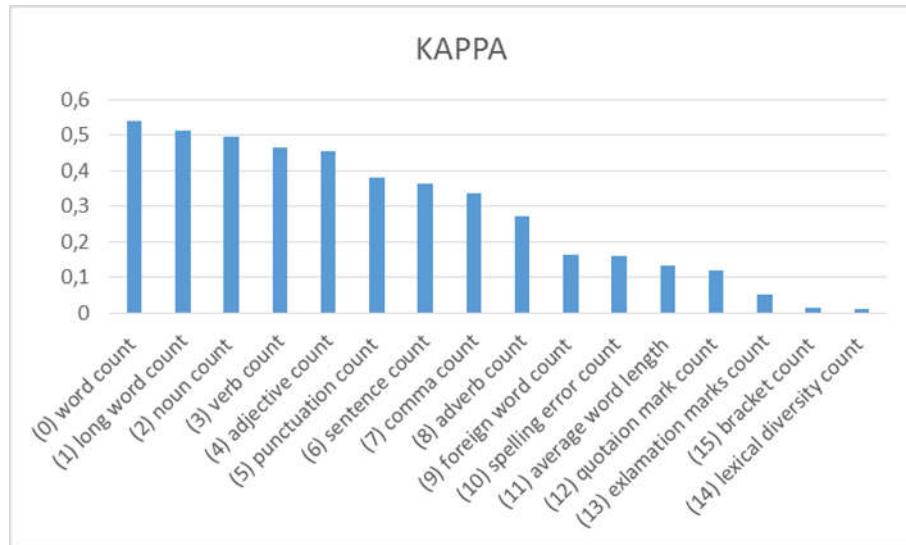
Gambar 22. Proses *forward feature* RMSE

Pada gambar 22 grafik yang warna merah adalah grafik yang menunjukkan bahwa penambahan suatu *feature* memiliki nilai RMSE yang lebih tinggi dibandingkan dengan nilai RMSE sebelumnya. Dan *feature* akhir dalam proses ini adalah [0,1,2,3,4,6,8,9,11,14,15] yaitu *word count*, *long word count*, *noun count*, *verb count*, *punctuation count*, *sentence count*, *comma count*, *spelling error count*, *average word count*, *lexical diversity count*, dan *bracket count*.



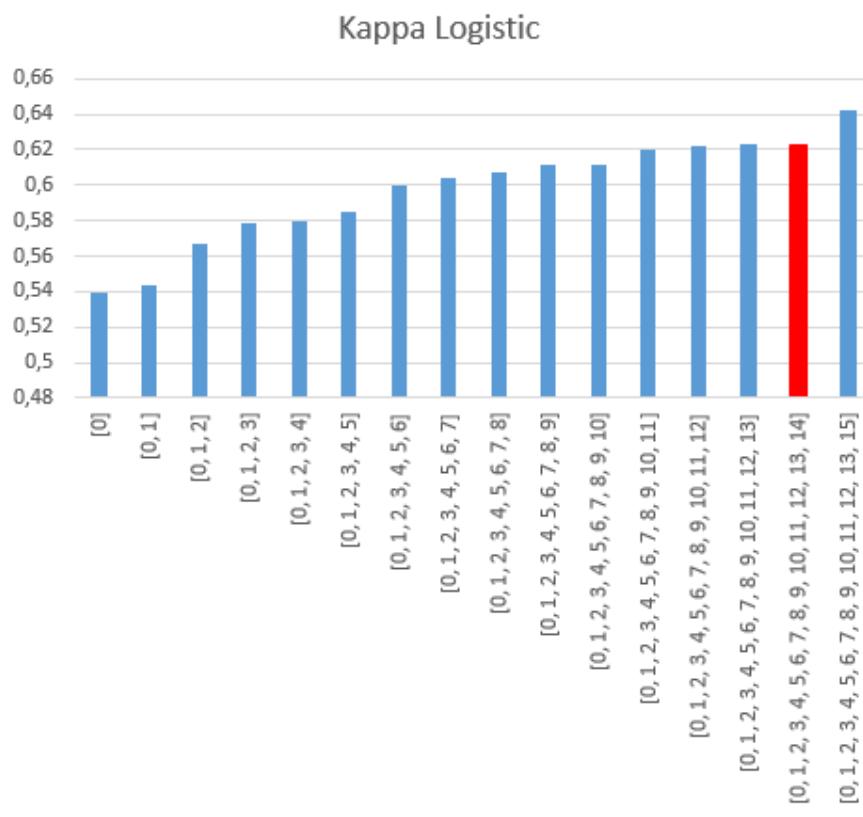
### 5.3.3 Logistic Regression Model Measured by Kappa Score

Pada tahap ini proses yang dilakukan sama hal nya dengan proses perhitungan pada kappa score regresi *linear* pada sub bab sebelumnya yaitu 5.3.1 perbedaannya adalah pada tahap ini dilakukan dengan regresi *logistic*. Hasil dari perhitungan dapat dilihat pada gambar 23.



Gambar 23. Perhitungan pada kappa score regresi logistic

Pada gambar 23 dapat dilihat bahwa *feature word count* memiliki nilai dengan tingkat akurasi yang tertinggi dan hasil dari pada *forward selection* dapat dilihat pada gambar 24.

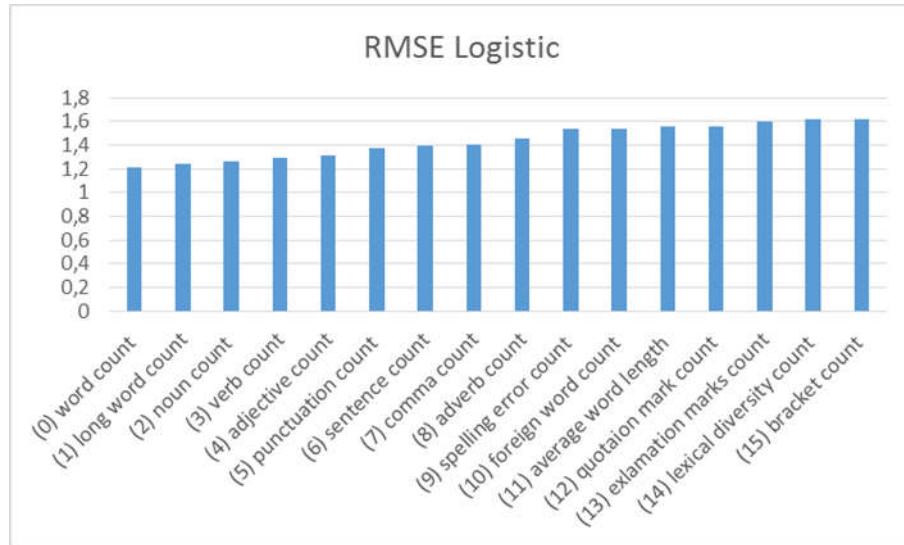


Gambar 24. Hasil *forward selection*

Pada gambar 24 grafik yang warna merah adalah grafik yang menunjukkan bahwa penambahan suatu *feature* memiliki nilai kappa yang lebih rendah dibandingkan dengan nilai kappa sebelumnya. Dan *feature* akhir dalam proses ini adalah [0,1,2,3,4,5,6,7,8,9,10,11,12,15] yaitu *word count, long word count, noun count, verb count, adjective count, punctuation count, sentence count, comma count, adverb count, foreign word count, spelling error count, average word count, quotation word count, exclamation word count dan bracket count.*

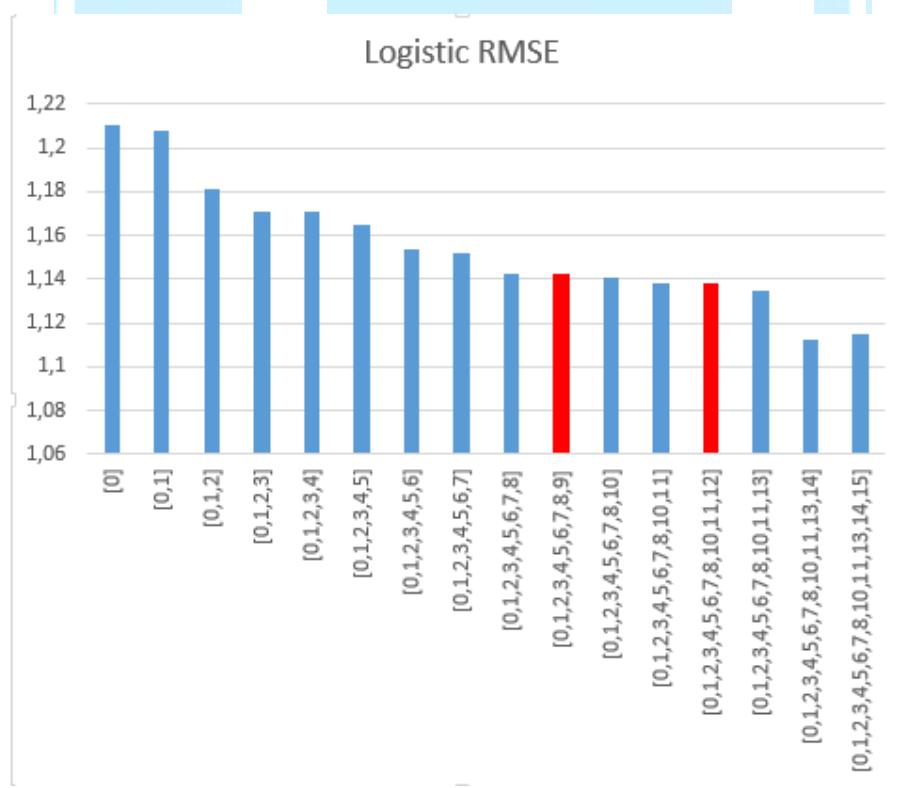
### 5.3.4 Logistic Regression Model Measured by RMSE Score

Pada tahap ini proses yang dilakukan sama hal nya dengan proses perhitungan pada RMSE score regresi *linear* pada sub bab sebelumnya yaitu 5.3.2 perbedaannya adalah pada tahap ini dilakukan dengan regresi *logistic*. Hasil dari perhitungan dapat dilihat pada gambar 25.



Gambar 25. Perhitungan pada RMSE score regresi logistic

Pada gambar 25 dapat dilihat bahwa *feature word count* memiliki nilai RMSE yang terendah dan hasil dari pada *forward selection* dapat dilihat pada gambar 26.



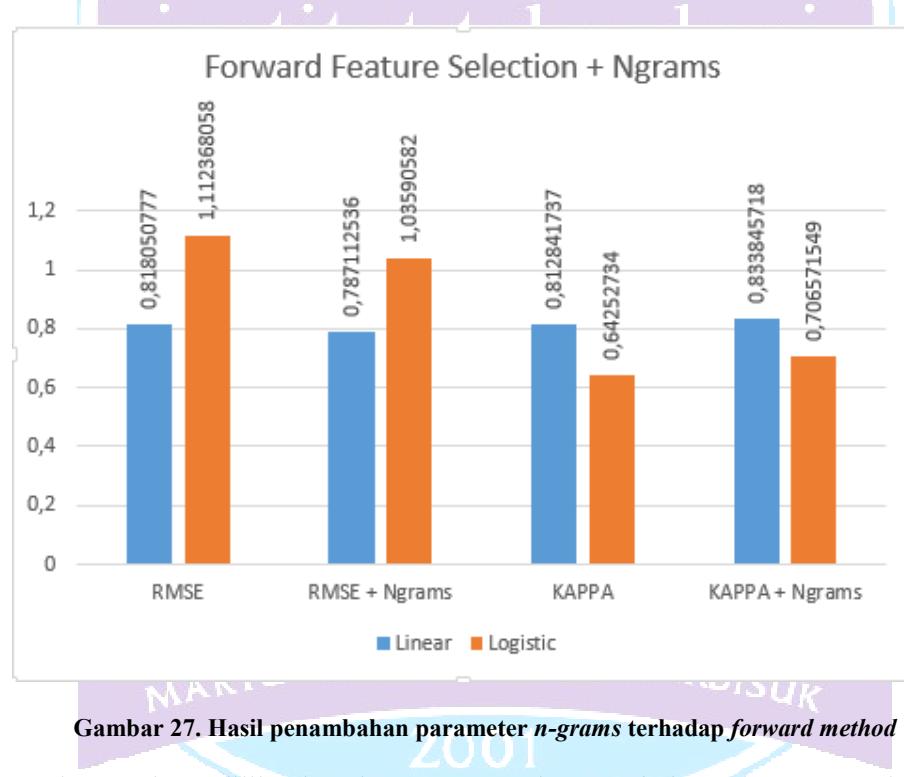
Gambar 26. Hasil dari pada forward selection

Pada gambar 26 grafik yang warna merah adalah grafik yang menunjukkan bahwa penambahan suatu *feature* memiliki nilai RMSE yang lebih tinggi dibandingkan dengan

nilai RMSE sebelumnya. Dan *feature* akhir dalam proses ini adalah [0,1,2,3,4,5,6,7,8,10,11,13,14,15] yaitu *word count*, *long word count*, *noun count*, *verb count*, *adjective count*, *punctuation count*, *sentence count*, *comma count*, *adverb count*, *foreign word count*, *average word count*, *exclamation word count*, *lexical diversity count* dan *bracket count*.

#### 5.4 Penambahan Parameter N-grams terhadap Forward Method

Penambahan *n-grams* untuk membuktikan apakah *n-grams* memiliki hasil dengan tingkat akurasi yang lebih tinggi atau memiliki nilai RMSE yang lebih rendah dibandingkan dengan tidak adanya penambahan *n-grams*. Untuk hasilnya dapat dilihat pada gambar 27.

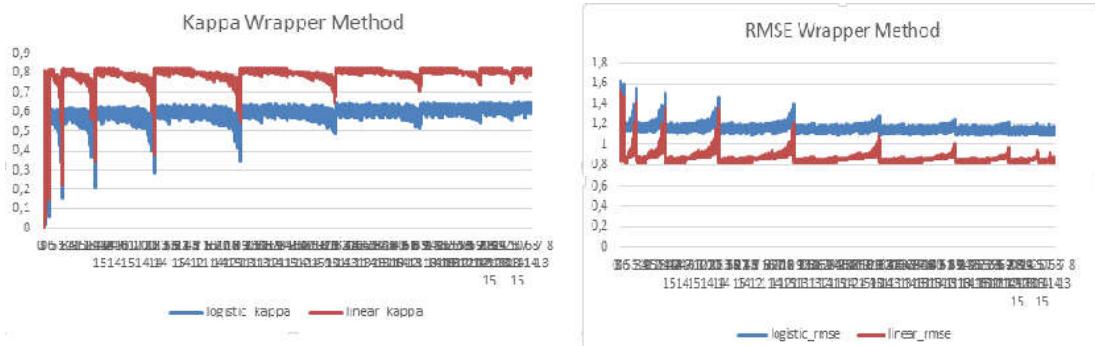


Gambar 27. Hasil penambahan parameter *n-grams* terhadap forward method

Pada gambar 27 dapat dilihat bagaimana pengaruh penambahan *N-grams* untuk pemilihan *features*, terlihat bahwa *n-grams feature* mampu menaikkan nilai tingkat akurasi pada perhitungan kappa dan menurunkan nilai RMSE baik itu *logistic* atau *linear*.

#### 5.5 Wrapper Features Selection

*Wrapper method* pada *feature selection* dilakukan dengan melakukan perhitungan terhadap subset dari ke 16 *feature*, sebanyak 65536 *subset feature* yang terbentuk dan hasil dari pada proses ini dapat dilihat pada gambar 28.

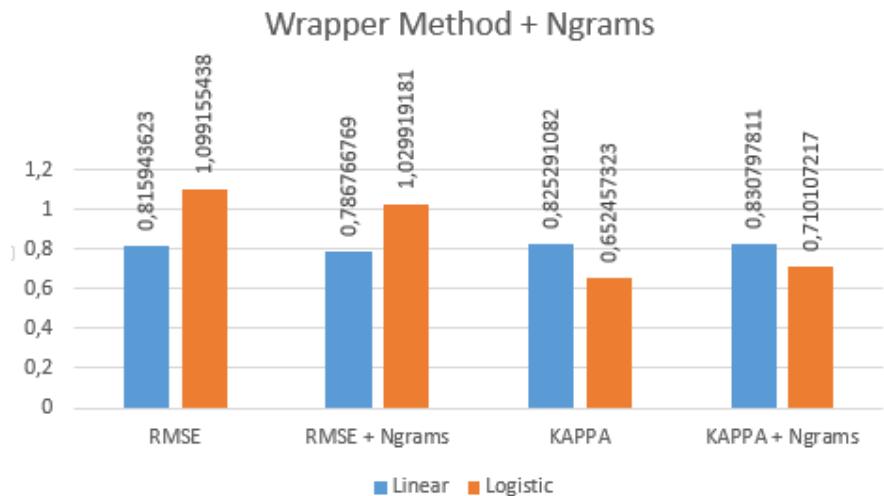


**Gambar 28. Proses wrapper features selection**

Pada diagram diatas dapat dilihat bahwa metode regresi *linear* memiliki grafik nilai tingkat akurasi kappa yang tinggi dan memiliki nilai grafik RMSE yang rendah dibandingkan dengan regresi *logistic*. Itu membuktikan bahwa perhitungan dengan menggunakan regresi *linear* jauh lebih baik dibandingkan dengan *logistic*. Dalam proses ini didapat hasil dengan tingkat akurasi *logistic* kappa sebesar 0,65245 dan *linear* kappa sebesar 0,8252 dengan hasil *feature* dari *logistic* adalah *word count*, *long word count*, *adjective count*, *punctuation count*, *comma count*, *spelling error count*, *foreign word count*, *average word count*, *exlamation marks count* dan *lexical diversity count*. Dan *feature* hasil dari *linear* adalah *word count*, *long word count*, *noun count*, *verb count*, *adjective count*, *sentence count*, *adverb count*, *spelling error count*, *average word count* dan *lexical diversity count*.

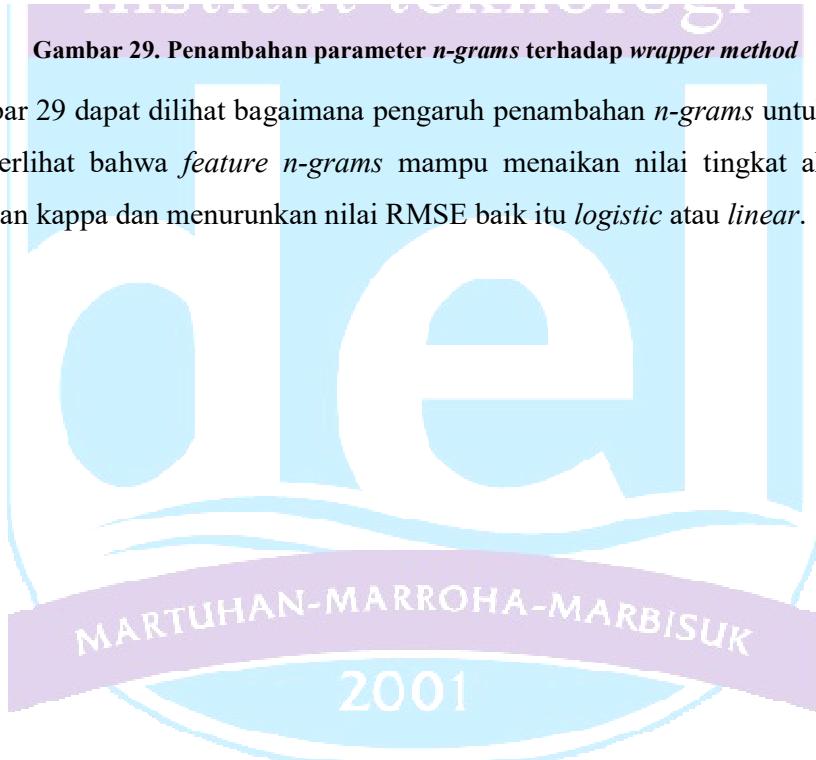
## 5.6 Penambahan Parameter N-grams terhadap Wrapper Method

Penambahan *n-grams* untuk membuktikan apakah *n-grams* memiliki hasil dengan tingkat akurasi yang lebih tinggi atau memiliki nilai RMSE yang lebih rendah dibandingkan dengan tidak adanya penambahan *n-grams* dalam menggunakan *wrapper method* dalam *feature selection*. Untuk hasilnya dapat dilihat pada gambar 29.



**Gambar 29. Penambahan parameter *n-grams* terhadap *wrapper method***

Pada gambar 29 dapat dilihat bagaimana pengaruh penambahan *n-grams* untuk pemilihan *features*, terlihat bahwa *feature n-grams* mampu menaikkan nilai tingkat akurasi pada perhitungan kappa dan menurunkan nilai RMSE baik itu *logistic* atau *linear*.



## Bab VI Kesimpulan dan Saran

Bab ini menjelaskan kesimpulan dari eksperimen dan saran yang didapatkan selama pengerjaan Tugas Akhir.

### 6.1 Kesimpulan

1. Pada Tugas Akhir ini, dihasilkannya tingkat akurasi prediksi skor esai yang lebih baik dengan menggunakan *wrapper method*, dengan dilakukannya *wrapper method* dihasilkan tingkat akurasi 0,825 untuk model *linear* dan 0,652 untuk *logistic*.
2. Model *multivariate linear regression* dapat bekerja dengan baik untuk *Automated Essay* dibuktikan dengan hasil tingkat akurasi yang tinggi dengan menggunakan model tersebut dibandingkan dengan model *logistic*.
3. Dengan adanya penambahan *n-grams feature* dicapai nya tingkat akurasi model prediktif yang lebih baik. Penambahan *n-grams* untuk metode *wrapper* dapat meningkatkan tingkat akurasi dari 0,825 menjadi 0,830

### 6.2 Saran

1. Penulis berasumsi bahwa *Feature Selection* menggunakan *wrapper method* dengan teknik *heuristic* untuk optimasi dalam proses pencarian dapat mengoptimalkan fungsi pencarian *features*.
2. Penulis tidak mencoba opsi lain untuk *feature engineering* yakni *dimensionality reduction* seperti *principal component analysis* (PCA). Penulis percaya bahwa opsi tersebut patut dicoba untuk mengetahui korelasi penggunaan metode tersebut, terhadap akurasi prediksi model.
3. Penulis yakin bahwa dengan melakukan analisis korelasi dari setiap kombinasi *features* menghasilkan model prediktif yang lebih akurat.

## Daftar Pustaka

Alexopoulos, E. C., n.d. Introduction to Multivariate Regression Analysis. *PMC*, Volume 14.

Anon., 2005. *Kamus Besar Bahasa Indonesia*. s.l.:s.n.

Berry, M. W. & Kogan, J., 2010. Text Mining Application and Theory. *United Kingdom*.

Callear, D., J.-S., Jennifer & Soh, V., 2001. Bridging Gaps in Computerised Assessment of Texts. *International Conference on Advanced Learning Techniques*.

Chai, T. & Draxler, R. R., 2012. Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. In *Geoscientific Model Development*, p. 1248.

DIKLI, S., 2006. Automated Essay Scoring. *Distance Education-TODJE*, Volume 7.

DIKLI, S., 2006. Automated Essay Scoring. *Distance Essay Scoring*.

Guyon & Eliseff, 2006. An Introduction to Feature Extraction.

Guyon, I. & Eliseff, A., 2003. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, pp. 1157-1182.

Kaggle, 2012. *The Hewlett Foundation: Automated Essay Scoring*. [Online] Available at: <https://www.kaggle.com/c/asap-aes>

Kurniawan, B., Effendi, S. & Sitompul, O. S., 2012. Klasifikasi Konten Berita Dengan Metode Text Mining. *DUNIA TEKNOLOGI INFORMASI*, Volume 1, p. 1.

Kurniawan, D., 2008. REGRESI LINIER (LINEAR REGRESSION).

Lauder, et al., 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, pp. 391-407.

Leidiyana, H., 2013. PENERAPAN ALGORITMA K-NEAREST NEIGHBOR UNTUK PENENTUAN RESIKO KREDIT KEPEMILIKAN KENDARAAN BEMOTOR. *Jurnal Penelitian Ilmu Komputer, System Embedded & Logic*, pp. 66-76.

Machine, J., 1990. *Python Software Foundation*. [Online] Available at: <https://pypi.python.org/pypi/xlrd>

Mahana, M., Johns, M. & Apte, A., 2012. Automated Essay Grading Using Machine.

Mitchell, T., Russell, T., Broomhead, P. & Aldridge, N., 2002. Towards Robust Computerised Marking of. *Proceedings of the Sixth International Computer Assisted Assessment Conference*.

Ng, A., 2002. *CS 229 Machine Learning Autumn 20012 Lecture Notes*. [Online]  
Available at: <http://cs229.stanford.edu/materials.html>

O., 2017. *Json/xml to Exel Online Conversion*. [Online]  
Available at: <http://www.json-xls.com>

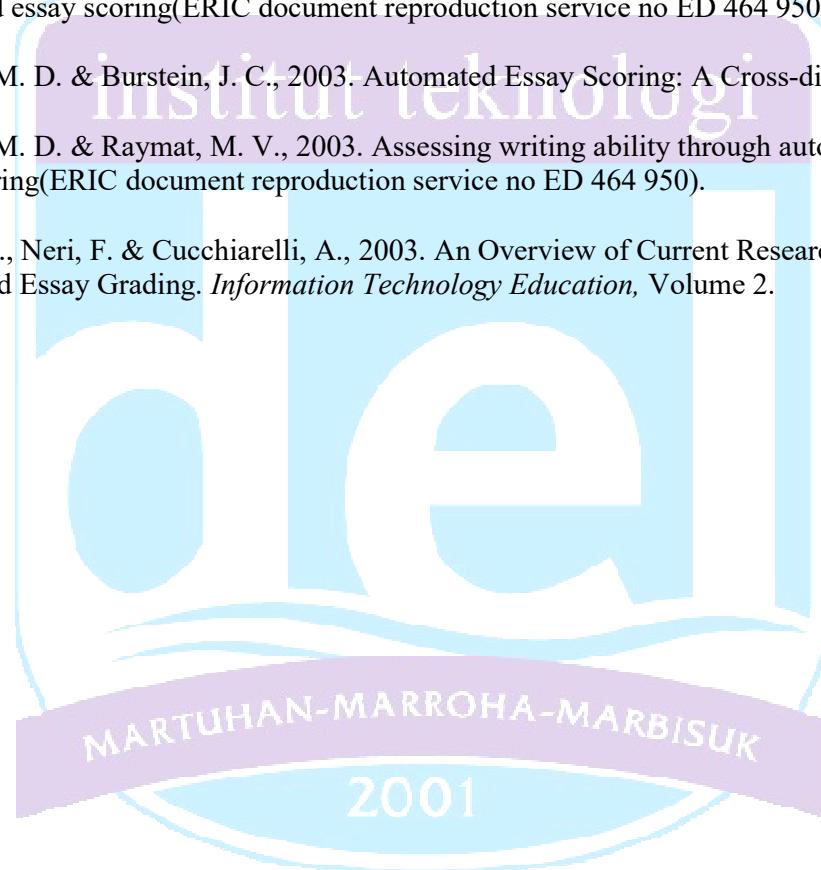
Riandyanai, D. A., P., Ketut, G. D. & Buana, P. W., 2014. Comparizing Fuzzy Logic and Fuzzy C-Means (FCM) on Summarizing Indonesian Language Document. *Journal of Theoretical and Applied Information Technology*, Volume 59.

Shermis, M. D. & Barrera, F. D., 2002. Exit assesments:evaluating writing ability through automated essay scoring(ERIC document reproduction service no ED 464 950).

Shermis, M. D. & Burstein, J. C., 2003. Automated Essay Scoring: A Cross-disciplianary.

Shermis, M. D. & Raymat, M. V., 2003. Assessing writing ability through automated essay scoring(ERIC document reproduction service no ED 464 950).

Valenti, S., Neri, F. & Cucchiarelli, A., 2003. An Overview of Current Research on Automated Essay Grading. *Information Technology Education*, Volume 2.



# Lampiran

## Lampiran 1

Pada lampiran 1 terdapat hasil dari *training model*:

```
In [8]: trainAndMeasureLogisticRegressionSingleFeature(X, Y)
Out[8]:
{'kappaList': OrderedDict([('word_count', 0.53971464526933155),
 ('long_word_count', 0.5141519110554581),
 ('noun_count', 0.4959179478779393),
 ('verb_count', 0.46366033257495587),
 ('adjective_count', 0.45557587520744292),
 ('punctuation_count', 0.37893782660583719),
 ('sentence_count', 0.36285727062755746),
 ('comma_count', 0.33690328628100974),
 ('adverb_count', 0.27089316540525066),
 ('foreign_word_count', 0.16138487048394468),
 ('spelling_error_count', 0.1599538236187279),
 ('average_word_length', 0.13371176917642269),
 ('quotaiion_mark_count', 0.11914815838239123),
 ('exlamation_marks_count', 0.050843784356833122),
 ('bracket_count', 0.012755297654190034),
 ('lexical_diversity_count', 0.01104570254321624)]),
 'rmseList': OrderedDict([('word_count', 1.2106839928983553),
 ('long_word_count', 1.2436498644930354),
 ('noun_count', 1.267765181083391),
 ('verb_count', 1.2940993088288837),
 ('adjective_count', 1.3119183965893175),
 ('punctuation_count', 1.3720273005857166),
 ('sentence_count', 1.3911623669004745),
 ('comma_count', 1.4022674179202705),
 ('adverb_count', 1.4566676660520463),
 ('spelling_error_count', 1.5367064023749757),
 ('foreign_word_count', 1.5415870690422391),
 ('average_word_length', 1.5619016746989374),
 ('quotaiion_mark_count', 1.5621044725272051),
 ('exlamation_marks_count', 1.6022992968501804),
 ('lexical_diversity_count', 1.6184294184762529),
 ('bracket_count', 1.6238414272862607)])}
```



---

```
In [9]: trainAndMeasureLinearRegressionSingleFeature(X, Y)
Out[9]:
{'kappaList': OrderedDict([('word_count', 0.78398691626255623),
 ('noun_count', 0.75228127145121293),
 ('long_word_count', 0.75021703645751181),
 ('verb_count', 0.70099446663247167),
 ('punctuation_count', 0.67298127117389184),
 ('adjective_count', 0.67229501865055252),
 ('sentence_count', 0.64553626408495413),
 ('comma_count', 0.53978430934023891),
 ('adverb_count', 0.53815423839505194),
 ('spelling_error_count', 0.32073620528369629),
 ('average_word_length', 0.301999576365097),
 ('foreign_word_count', 0.29758647911504588),
 ('quotaiton_mark_count', 0.19083882111776052),
 ('exlamation_marks_count', 0.16858626448992053),
 ('lexical_diversity_count', 0.043191717867844925),
 ('bracket_count', 0.0048961268696924298)]),
 'rmseList': OrderedDict([('word_count', 0.88296955122662735),
 ('long_word_count', 0.94778763786107911),
 ('noun_count', 0.95252483683027511),
 ('verb_count', 1.0215205074535096),
 ('punctuation_count', 1.0610956784425156),
 ('adjective_count', 1.0630636420364419),
 ('sentence_count', 1.1058966198777838),
 ('adverb_count', 1.2045799379716586),
 ('comma_count', 1.2233660528584898),
 ('spelling_error_count', 1.3769453623722125),
 ('foreign_word_count', 1.3819306229746777),
 ('average_word_length', 1.3820113800660632),
 ('quotaiton_mark_count', 1.4739212251507934),
 ('exlamation_marks_count', 1.490370744833827),
 ('lexical_diversity_count', 1.5262787320472526),
 ('bracket_count', 1.5300417782636633)])}
```



## Lampiran 2

pada lampiran 2 terdapat hasil dari *forward features selection* :

```
In [51]: forwardFeatureSelectionRMSE(XRMSELinear,Y,'linear')
rmse = 0.882969551227
1 --> [0, 1]
rmse = 0.880525831685
2 --> [0, 1, 2]
rmse = 0.860691877672
3 --> [0, 1, 2, 3]
rmse = 0.85807571677
4 --> [0, 1, 2, 3, 4]
rmse = 0.847888553976
5 --> [0, 1, 2, 3, 4, 5]
6 --> [0, 1, 2, 3, 4, 6]
rmse = 0.84746635014
7 --> [0, 1, 2, 3, 4, 6, 7]
8 --> [0, 1, 2, 3, 4, 6, 8]
rmse = 0.846288684316
9 --> [0, 1, 2, 3, 4, 6, 8, 9]
rmse = 0.845153566635
10 --> [0, 1, 2, 3, 4, 6, 8, 9, 10]
11 --> [0, 1, 2, 3, 4, 6, 8, 9, 11]
rmse = 0.839220160302
12 --> [0, 1, 2, 3, 4, 6, 8, 9, 11, 12]
13 --> [0, 1, 2, 3, 4, 6, 8, 9, 11, 13]
14 --> [0, 1, 2, 3, 4, 6, 8, 9, 11, 14]
rmse = 0.818050776887
15 --> [0, 1, 2, 3, 4, 6, 8, 9, 11, 14, 15]
Out[51]: [0, 1, 2, 3, 4, 6, 8, 9, 11, 14]
```

```
In [52]: trainAndMeasureLinearRegressionCrossValidate(XRMSELinearForwardNGRAMS, Y)
Out[52]: {'kappa': 0.83586036623864879, 'rmse': 0.7871125355042794}
```



```
In [49]: forwardFeatureSelectionKappa(XKappaLinear,Y,'linear')
kappa = 0.783986916263
1 --> [0, 1]
kappa = 0.796907472903
2 --> [0, 1, 2]
kappa = 0.798485269888
3 --> [0, 1, 2, 3]
4 --> [0, 1, 2, 4]
kappa = 0.808573024104
5 --> [0, 1, 2, 4, 5]
6 --> [0, 1, 2, 4, 6]
7 --> [0, 1, 2, 4, 7]
8 --> [0, 1, 2, 4, 8]
9 --> [0, 1, 2, 4, 9]
10 --> [0, 1, 2, 4, 10]
11 --> [0, 1, 2, 4, 11]
12 --> [0, 1, 2, 4, 12]
13 --> [0, 1, 2, 4, 13]
14 --> [0, 1, 2, 4, 14]
kappa = 0.812021039378
15 --> [0, 1, 2, 4, 14, 15]
kappa = 0.812841736932
Out[49]: [0, 1, 2, 4, 14, 15]
```

```
In [50]: trainAndMeasureLinearRegressionCrossValidate(XKappaLinearForwardNGRAMS, Y)
Out[50]: {'kappa': 0.83384571787465689, 'rmse': 0.7889927916458509}
```



```
In [75]: forwardFeatureSelectionKappa(XKappaLogistic,Y,'logistic')
kappa = 0.539714645269
1 --> [0, 1]
kappa = 0.543968071846
2 --> [0, 1, 2]
kappa = 0.566664252215
3 --> [0, 1, 2, 3]
kappa = 0.578204151657
4 --> [0, 1, 2, 3, 4]
kappa = 0.579177044834
5 --> [0, 1, 2, 3, 4, 5]
kappa = 0.585369349086
6 --> [0, 1, 2, 3, 4, 5, 6]
kappa = 0.599370060914
7 --> [0, 1, 2, 3, 4, 5, 6, 7]
kappa = 0.60366531472
8 --> [0, 1, 2, 3, 4, 5, 6, 7, 8]
kappa = 0.60759259702
9 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
kappa = 0.610929779812
10 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
kappa = 0.611416594013
11 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
kappa = 0.620374121058
12 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
kappa = 0.622259976205
13 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
kappa = 0.623033454618
14 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
15 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15]
kappa = 0.642527340456
Out[75]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15]

In [76]: trainAndMeasureLogisticRegressionCrossValidate(XKappaLogisticFor
Out[76]: {'kappa': 0.70657154934170274, 'rmse': 1.0419157563164301}

In [77]: |
```



```

In [73]: forwardFeatureSelectionRMSE(XRMSELogistic,Y,'logistic')
rmse = 1.2106839929
1 --> [0, 1]
rmse = 1.2078703291
2 --> [0, 1, 2]
rmse = 1.18155794218
3 --> [0, 1, 2, 3]
rmse = 1.17110935021
4 --> [0, 1, 2, 3, 4]
rmse = 1.17049274234
5 --> [0, 1, 2, 3, 4, 5]
rmse = 1.16467443284
6 --> [0, 1, 2, 3, 4, 5, 6]
rmse = 1.15335408075
7 --> [0, 1, 2, 3, 4, 5, 6, 7]
rmse = 1.1521350747
8 --> [0, 1, 2, 3, 4, 5, 6, 7, 8]
rmse = 1.14217317393
9 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
10 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 10]
rmse = 1.14098065138
11 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11]
rmse = 1.13503685997
12 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12]
13 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13]
rmse = 1.13461192145
14 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13, 14]
rmse = 1.11236805832
15 --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13, 14, 15]
Out[73]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13, 14]

In [74]: trainAndMeasureLogisticRegressionCrossValidate(XRMSELogisticForwardNGRAMS, Y)
Out[74]: {'kappa': 0.70707624554541781, 'rmse': 1.0359058201268394}

```

### Lampiran 3

pada lampiran 3 terdapat hasil dari *wrapper features selection* :



```
In [122]:  
trainAndMeasureLinearRegressionCrossValidate(XRMSELinearWrapperNGRAMS, Y)  
Out[122]: {'kappa': 0.83453090095683669, 'rmse': 0.78676676854392791}  
  
In [123]:  
trainAndMeasureLogisticRegressionCrossValidate(XRMSELogisticWrapperNGRAMS, Y)  
Out[123]: {'kappa': 0.71010721748693151, 'rmse': 1.0299191810388195}  
  
In [124]:  
trainAndMeasureLinearRegressionCrossValidate(XKappaLinearWrapperNGRAMS, Y)  
Out[124]: {'kappa': 0.83079781103169248, 'rmse': 0.78884926880877904}  
  
In [125]:  
trainAndMeasureLogisticRegressionCrossValidate(XKappaLogisticWrapperNGRAMS, Y)  
Out[125]: {'kappa': 0.71010721748693151, 'rmse': 1.0299191810388195}  
  
In [126]: |
```

```
In [122]:  
trainAndMeasureLinearRegressionCrossValidate(XRMSELinearWrapperNGRAMS, Y)  
Out[122]: {'kappa': 0.83453090095683669, 'rmse': 0.78676676854392791}
```

```
In [123]:  
trainAndMeasureLogisticRegressionCrossValidate(XRMSELogisticWrapperNGRAMS, Y)  
Out[123]: {'kappa': 0.71010721748693151, 'rmse': 1.0299191810388195}
```

```
In [124]:  
trainAndMeasureLinearRegressionCrossValidate(XKappaLinearWrapperNGRAMS, Y)  
Out[124]: {'kappa': 0.83079781103169248, 'rmse': 0.78884926880877904}
```

```
In [125]:  
trainAndMeasureLogisticRegressionCrossValidate(XKappaLogisticWrapperNGRAMS, Y)  
Out[125]: {'kappa': 0.71010721748693151, 'rmse': 1.0299191810388195}
```

```
In [126]: |
```



```
In [112]: wrapper['logistic_rmse'].min()
Out[112]: 1.0991554379999999

In [113]: wrapper['logistic_rmse'].argmin()
Out[113]: 63838

In [114]: wrapper['feature'][63838]
Out[114]: '0 1 3 4 5 7 9 10 11 12 13 14'

In [115]: wrapper['linear_rmse'].min()
Out[115]: 0.8159436230000001

In [116]: wrapper['linear_rmse'].argmin()
Out[116]: 51238

In [117]: wrapper['feature'][51238]
Out[117]: '0 1 2 3 5 7 8 9 11 15'

In [105]: wrapper['logistic_kappa'].max()
Out[105]: 0.6524573229999992

In [106]: wrapper['logistic_kappa'].argmax()
Out[106]: 63838

In [107]: wrapper['feature'][63838]
Out[107]: '0 1 3 4 5 7 9 10 11 12 13 14'

In [108]: wrapper['linear_kappa'].max()
Out[108]: 0.8252910820000001

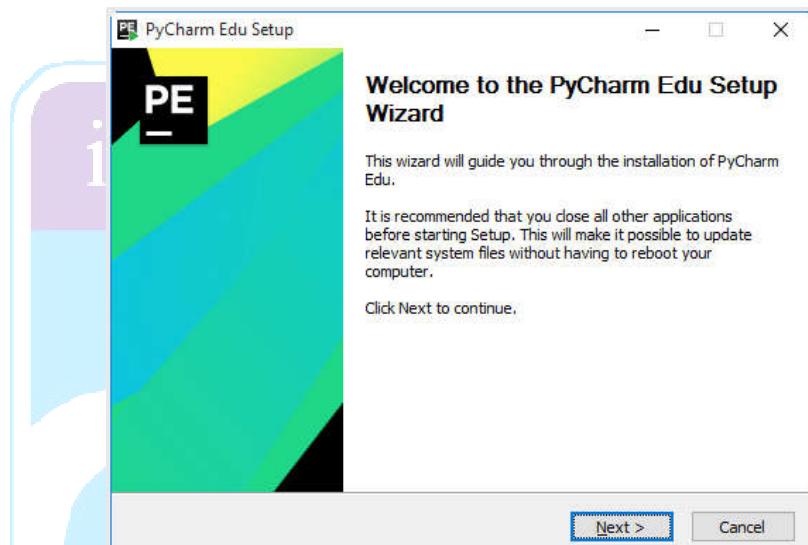
In [109]: wrapper['linear_kappa'].argmax()
Out[109]: 50915

In [110]: wrapper['feature'][50915]
Out[110]: '0 1 2 3 4 6 8 9 11 14'
```

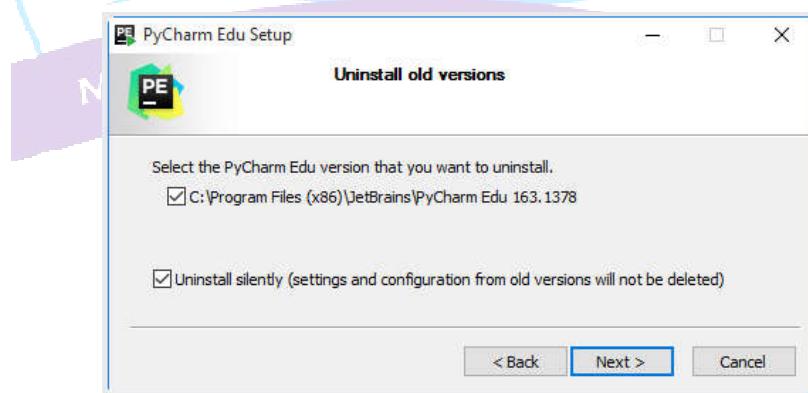


#### Lampiran 4 Pycharm Instalation Guide

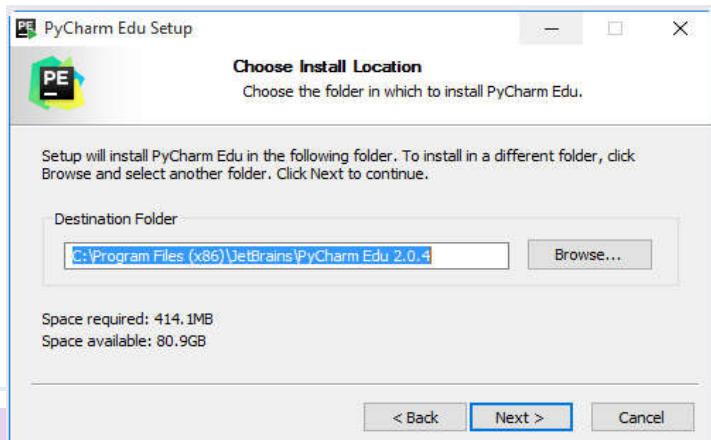
1. Unduh dari *website* resmi Jetbrains di <https://www.jetbrains.com/pycharm-edu/download>
2. *Double-Click file* pyCharmEDU-xxx.xxx.exe untuk menjalankan *installer*. Verifikasi dan buka halaman pertama *wizard* penginstalan dan klik *Next* dan klik *Next* untuk melanjutkan.



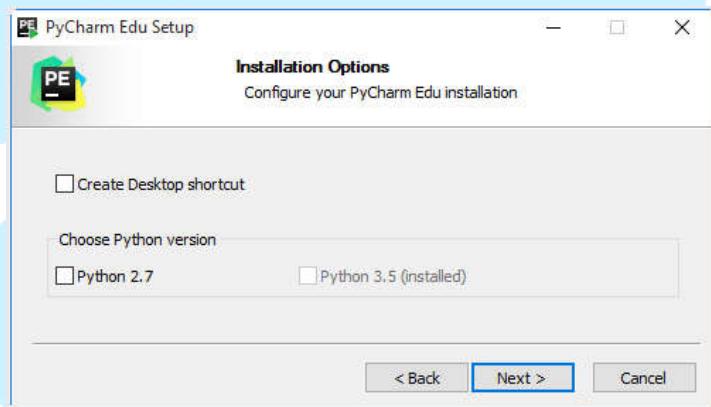
3. Pada halaman kedua *wizard*, Anda harus menentukan apakah Anda menginginkan instalasi sebelumnya dilepas pemasangannya dan klik *Next* untuk melanjutkan.



4. Pada layar *Choose Install Location*, tentukan direktori tujuan tempat produk akan dipasang. *Wizard* penginstalan menyarankan lokasi *default*. Untuk memilih lokasi khusus, klik *Browse* dan pilih lokasi yang diinginkan dalam sistem *file* Anda. Bila sudah siap, klik *Next*.



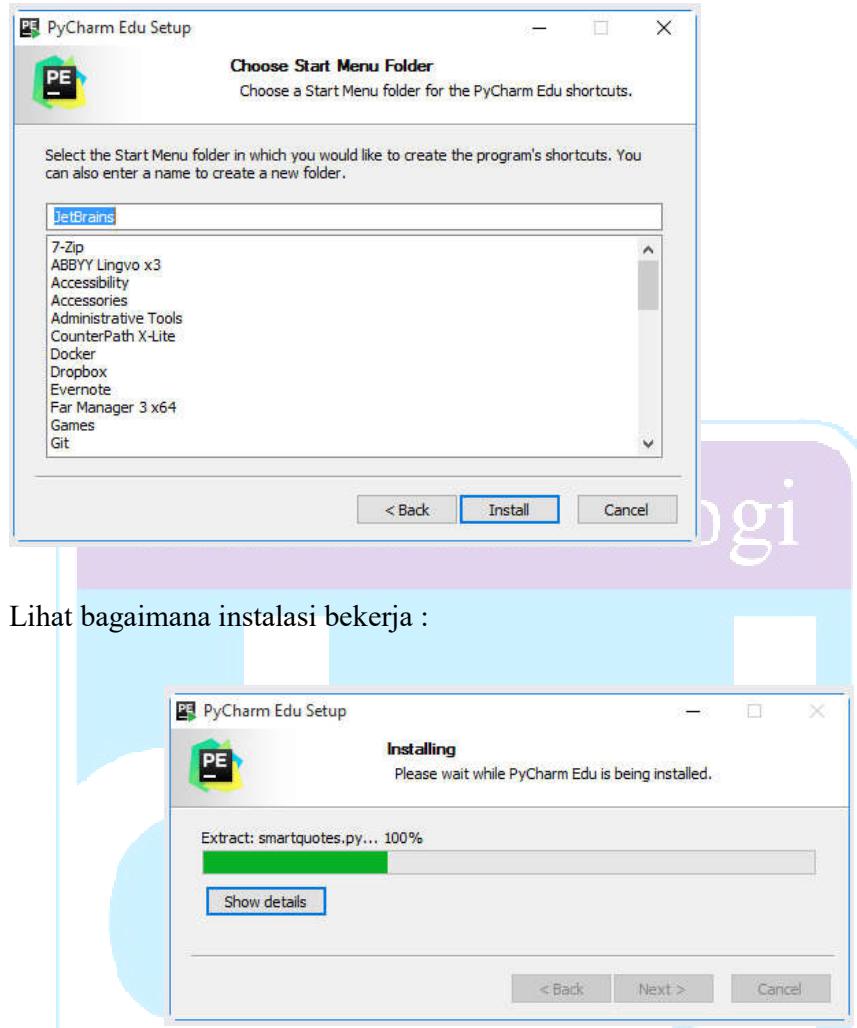
5. Pada layar *Installation Options*, tentukan pilihan berikut:



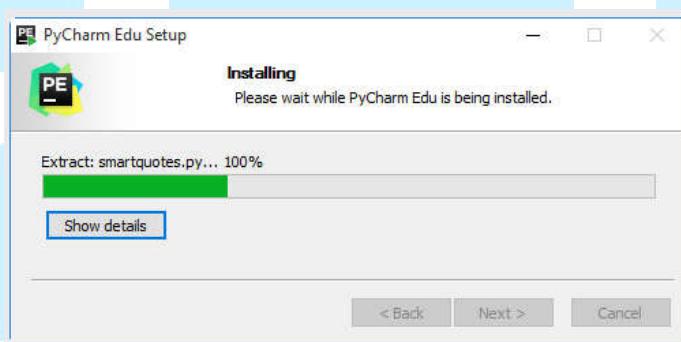
**Create shortcut Desktop:** Jika Anda memilih kotak centang ini, jalan pintas ke PyCharm Edu akan muncul di *Desktop* Anda. Dengan demikian Anda akan bisa meluncurkan PyCharm Edu dari *Desktop* Anda.

**Choose Python version:** Pada bagian ini, Anda harus menentukan versi *Python* yang akan diinstal untuk PyCharm Edu. *Installer* menyarankan Anda untuk memilih antara *Python 2.7* dan *Python 3.5* (yang sudah terpasang). Jika Anda tidak yakin versi mana yang cocok untuk Anda, lihat perbandingan versinya.

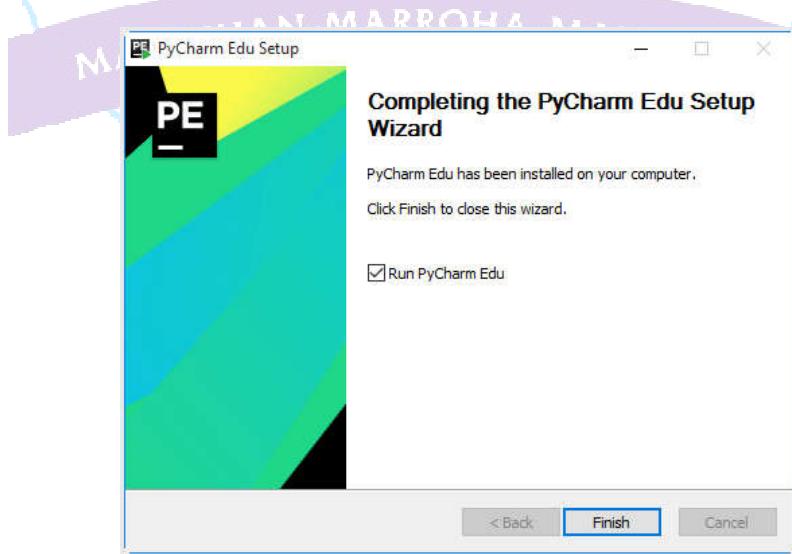
6. Di halaman *wizard* penginstalan ini, Anda harus menentukan di *folder* mana dari menu *Start*, *shortcut* aplikasi akan muncul. Jika Anda tidak memasukkan nama *folder*, maka menu *Start* Anda akan berisi *folder* JetBrains. Klik *Install*.



Lihat bagaimana instalasi bekerja :

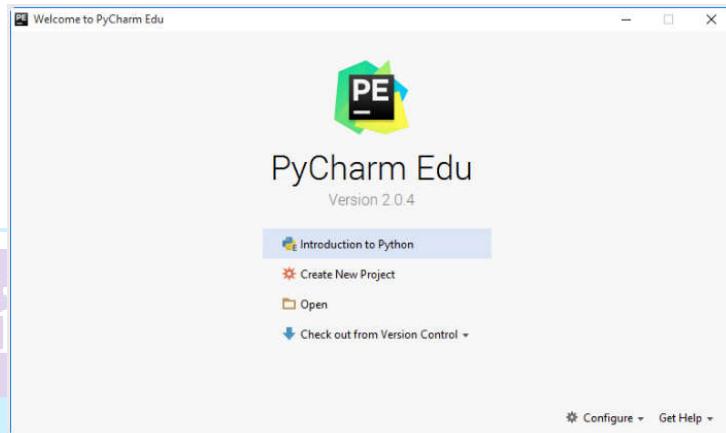


7. Saat penginstalan sudah siap, Anda akan melihat halaman terakhir *wizard*:



Pilih kotak centang Jalankan PyCharm Edu (jika Anda ingin segera meluncurkan produk), dan klik *Finish*.

#### 8. First Launch



Jika Anda mengklik *Configure*, PyCharm menunjukkan daftar *dropdown* yang menyarankan Anda untuk mengkonfigurasi pengaturan/preferensi, plugin, pengaturan impor dan ekspor, dan memeriksa pembaruan.

## Lampiran 5 Source Code

```
1.      import math as mt
2.      from nltk.corpus import stopwords
3.      import xlrd
4.      import xlsxwriter
5.      import re
6.      import nltk
7.      import enchant
8.      import numpy as np
9.      import pandas as pd
10.     import matplotlib.pyplot as plt
11.     from sklearn import linear_model, datasets
12.     from sklearn.metrics import cohen_kappa_score
13.     from sklearn.linear_model import LogisticRegression
14.     from sklearn.cross_validation import KFold
15.     from sklearn.cross_validation import cross_val_score
16.     from sklearn.model_selection import cross_val_predict
17.     from sklearn import metrics
18.     from sklearn import preprocessing
19.     from collections import OrderedDict
20.
21. """
22. ##### IMPORT DATASET #####
23. # DATASET LOAD, FEATURES MATRIX, N-GRAMS MATRIX, OUTPUT VECTOR
24. #####
25. """
26. data = pd.read_csv('C:\Users\hikennoace\Desktop\SIDANG TUGAS AKHIR\X.csv')
27. target = pd.read_csv('C:\Users\hikennoace\Desktop\SIDANG TUGAS AKHIR\Y.csv')
28. X = np.asarray(data)
29. Y = np.asarray(target)
30. Y = Y.reshape((1783L,))
31. NGrams= np.asarray(pd.read_csv('C:\Users\hikennoace\Desktop\SIDANG TUGAS
AKHIR\NGRAMS.csv'))
32. XKappaLinear = np.asarray(pd.read_csv('C:\Users\hikennoace\Desktop\SIDANG TUGAS
AKHIR\XSortByKappaLinear.csv'))
33. XKappaLinear = XKappaLinear[:,1:]
```

```

34.     XKappaLogistic = np.asarray(pd.read_csv('C:\Users\hikennoace\Desktop\SIDANG  
TUGAS AKHIR\XSortByKappaLogistic.csv'))
35.     XKappaLogistic = XKappaLogistic[:,1:]
36.     XRMSELinear = np.asarray(pd.read_csv('C:\Users\hikennoace\Desktop\SIDANG TUGAS  
AKHIR\XSortByRMSELinear.csv'))
37.     XRMSELinear = XRMSELinear[:,1:]
38.     XRMSELogistic =np.asarray(pd.read_csv('C:\Users\hikennoace\Desktop\SIDANG TUGAS  
AKHIR\XSortByRMSELogistic.csv'))
39.     XRMSELogistic = XRMSELogistic[:,1:]
40.     wrapper = pd.read_csv('C:\Users\hikennoace\Desktop\SIDANG TUGAS  
AKHIR\WrapperBruteForceMethod.csv')
41.     wrapperMethodResult =np.asarray(pd.read_csv('C:\Users\hikennoace\Desktop\SIDANG  
TUGAS AKHIR\WrapperBruteForceMethod.csv'))
42.
43.     XKappaLinearForwardNGRAMS = np.concatenate((XKappaLinear[:,[0, 1, 2, 4, 14, 15]],  
        NGrams),axis=1)
44.     XKappaLogisticForwardNGRAMS = np.concatenate((XKappaLogistic[:,[0, 1, 2, 3, 4, 5,  
        6, 7, 8, 9, 10, 11, 12, 13, 15]],NGrams),axis=1)
45.     XRMSELinearForwardNGRAMS = np.concatenate((XRMSELinear[:,[0, 1, 2, 3, 4, 6, 8, 9,  
        11, 14]],NGrams),axis=1)
46.     XRMSELogisticForwardNGRAMS = np.concatenate((XRMSELogistic[:,[0, 1, 2, 3, 4, 5,  
        6, 7, 8, 10, 11, 13, 14]],NGrams),axis=1)
47.
48.     XKappaLinearWrapperNGRAMS = np.concatenate((X[:,[0, 1, 2, 3, 4, 6, 8, 9, 11, 14]  
        ],NGrams),axis=1)
49.     XKappaLogisticWrapperNGRAMS = np.concatenate((X[:,[0, 1, 3, 4, 5, 7, 9, 10, 11,  
        12, 13, 14]],NGrams),axis=1)
50.     XRMSELinearWrapperNGRAMS = np.concatenate((X[:,[0, 1, 2, 3, 5, 7, 8, 9, 11, 15]  
        ],NGrams),axis=1)
51.     XRMSELogisticWrapperNGRAMS = np.concatenate((X[:,[0, 1, 3, 4, 5, 7, 9, 10, 11, 1  
        2, 13, 14]],NGrams),axis=1)
52.
53.     """
54.     ##### PREPROCESSING & FEATURES EXTRACTION PART #####
55.             NAME CONVENTION START WITH PFX + FUNCTION NUMBER
56.     #####
57.     """

```

```

58.
59.     """
60.     PFX 01 --> SpellingReplacer(object)
61.     Class Utility for repair the mismatch spelling take the object of string that we
62.     want to repair, reutrn the correct word as a return value
63.     """
64.     class SpellingReplacer(object):
65.         def __init__(self, dict_name = 'en_GB', max_dist = 2):
66.             self.spell_dict = enchant.Dict(dict_name)
67.             self.max_dist = 2
68.
69.         def replace(self, word):
70.             if self.spell_dict.check(word):
71.                 return word
72.             suggestions = self.spell_dict.suggest(word)
73.             if suggestions and edit_distance(word, suggestions[0]) <= self.max_dist:
74.                 return suggestions[0]
75.             else:
76.                 return word
77.     """
78.     PFX 02 --> PreprocessingStopWordRemoval(ListString)
79.     Function to do stopword removal preprocessing , take the List of string that
80.     already tokenized as parameter an return listString witout stopword
81.     """
82.     def PreprocessingStopWordRemoval(ListString):
83.         ListString = stopwords.clean(ListString, "en");
84.         stringToken = ""
85.         for data in ListString:
86.             stringToken = stringToken + data + '\n'
87.         return stringToken;
88.
89.     """
90.     PFX 03 --> PreprocessingCaseFolding(string)
91.     Function to do stopword removal preprocessing , take the string as a parameter
92.     and return a string with that consists only lowercase
93.     """

```

```

94.
95.     def PreprocessingCaseFolding(string):
96.         return string.lower()
97.
98.     """
99.     PFX 04 --> PreprocessingTokenization(string)
100.    Function to do stopword removal preprocessing , take the string as a parameter
101.    return a tokenized string
102.    """
103.
104.    def PreprocessingTokenization(string):
105.        return re.findall(r"\w+", string)
106.
107.
108.    """
109.    PFX 05 --> def HeaderRow()
110.    Function to define the header of a excel file
111.    """
112.    def HeaderRow():
113.        string = ['word count','long word count','noun count','verb count','comma
114.                  count','punctuation count','sentence count','adjective
115.                  count','adverb count','lexical diversity count','quoation mark
116.                  count','word length','braket count','spelling error
117.                  count','exlamation marks count','foreign word count'];
118.
119.    return string
120.
121.    """
122.    PFX 06 --> FeatureExtractionLongWordCount(ListString)
123.    Function to count the number of Long word from a List of string
124.    """
125.    def FeatureExtractionLongWordCount(ListString):
126.        longWordCount = 0
127.        sumLengthWord = 0
128.        stringListUnique = set(ListString)
129.        for word in stringListUnique:
130.            sumLengthWord += len(word)
131.        averageWordLength = sumLengthWord/len(stringListUnique)

```

```

127.         for word in stringListUnique:
128.             if(len(word)>=averageWordLength):
129.                 longWordCount+=1
130.             return longWordCount
131.
132. """
133. PFX 07 --> FeatureExtractionWordLength(ListString)
134. Function to measure the average word Length from a List of string
135. """
136. def FeatureExtractionWordLength(ListString):
137.     sumLengthWord = 0
138.     stringListUnique = set(ListString)
139.     for word in stringListUnique:
140.         sumLengthWord+=len(word)
141.     return sumLengthWord/len(stringListUnique)
142.
143. """
144. PFX 08 --> FeatureExtractionLexicalDiversity(ListString)
145. Function to measure the Lexical diversity from a List of string
146. """
147. def FeatureExtractionLexicalDiversity(ListString):
148.     return '%.2f' % (len(set(ListString)) / len(ListString));
149.
150. """
151. PFX 09 --> FeatureExtractionWordCount(ListString)
152. Function to count the number of word from a List of string
153. """
154. def FeatureExtractionWordCount(ListString):
155.     return len(Counter(ListString))
156.
157. """
158. PFX 10 --> FeatureExtractionCommaCount(string)
159. Function to count the number of comma in a string
160. """
161. def FeatureExtractionCommaCount(string):
162.     return string.count(',')
163.

```

```

164.      """
165.      PFX 11 --> FeatureExtractionQuotationMark(string)
166.      Function to count the number of Quotation Mark in a string
167.      """
168.      def FeatureExtractionQuotationMark(string):
169.          string = re.findall(r'\"(.+?)\"',string)
170.          return len(string)
171.
172.      """
173.      PFX 12 --> FeatureExtractionBracketCount(string))
174.      Function to count the number of Bracket in a string
175.      """
176.      def FeatureExtractionBracketCount(string):
177.          string = re.findall(r'\((.+?)\)',string)
178.          return len(string)
179.
180.      """
181.      PFX 13 --> FeatureExtractionExclamationMarks(string)
182.      Function to count the number of ExclamationMark in a string
183.      """
184.      def FeatureExtractionExclamationMarks(string):
185.          return string.count('!')
186.
187.      """
188.      PFX 14 --> FeatureExtractionSentenceCount(string)
189.      Function to count the number of Sentence in essay
190.      """
191.      def FeatureExtractionSentenceCount(string):
192.          return len(re.split(r'[.!?]+', string))
193.
194.      """
195.      PFX 15 --> FeatureExtractionPunctuationCount(string)
196.      Function to count the number of Punctuation in essay
197.      """
198.      def FeatureExtractionPunctuationCount(string):
199.          counts = Counter(string);

```

```

200.         punctuation_counts = {k:
201.             v for k, v in counts.iteritems() if k in punctuation and k != '@'}
202.         count = 0
203.         for key, value in punctuation_counts.iteritems():
204.             count += value
205.
206.         """
207.     PFX 16 --> FeatureExtractionPartOfSpeechCount(ListString)
208.     Function to count number of Part of Speech Tagger in essay like noun, verb,
209.     adjective etc
210.     """
211.     def FeatureExtractionPartOfSpeechCount(ListString):
212.         data = nltk.pos_tag(ListString)
213.         counts = Counter(tag for word, tag in data)
214.         sumNoun = 0;
215.         sumVerb = 0;
216.         sumAdverb = 0;
217.         sumAdjective = 0;
218.         for key, value in counts.iteritems():
219.             if "NN" in key: # Noun has key begin with NN
220.                 sumNoun += value
221.             elif "VB" in key: # Verb has key begin with VB
222.                 sumVerb += value
223.             elif "RB" in key: # Adverb has key begin with RB
224.                 sumAdverb += value
225.             elif "JJ" in key: # Adjective has key begin with JJ
226.                 sumAdjective += value
227.         result = {"noun": sumNoun, "verb": sumVerb, "adverb":
228.             sumAdverb, "adjective":sumAdjective};
229.         return result
230.
231.         """
232.     PFX 17 --> FeatureExtractionSpellErrorCount(ListString)
233.     Function to count number of error spelling in essay
234.     """

```

```

234.     def FeatureExtractionSpellErrorCount(ListString) :
235.         count = 0
236.         for item in ListString:
237.             replacer = SpellingReplacer()
238.             r = replacer.replace(item)
239.             if(item!=r):
240.                 count+=1
241.         return count
242.
243. """
244. PFX 18 --> FeatureExtractionForeignWordCount(ListString)
245. Function to count numbers of Foreign Word spelling in essay
246. """
247. def FeatureExtractionForeignWordCount(ListString) :
248.     count = 0
249.     dictionary = enchant.Dict("en_US")
250.     for item in ListString:
251.         if(dictionary.check(item)==False):
252.             count+=1
253.     return count
254.
255. """
256. PFX 19 --> FeatureExtractionNGram(ListString,n)
257. Function to do feature extraxtion N-Grams
258. """
259. def FeatureExtractionNGram(ListString,n):
260.     ngram = ngrams(ListString, n)
261.     list = [];
262.     for grams in ngram:
263.         list.append(grams)
264.     return len(set(list))
265.
266. """
267. PFX 20 --> application(newNameExcelFile, LocationDataSetExcel, nameDataSetExcel)
268. Function to do all preprocessing and features extraction and record the result
269. to an excel file
269. """

```

```

270.
271.     def application(newNameExcelFile, locationDataSetExcel, nameDataSetExcel):
272.         workbook = xlsxwriter.Workbook(newNameExcelFile)
273.         worksheet = workbook.add_worksheet()
274.         fileExcel = join(dirname(dirname(abspath(__file__))), locationDataSetExcel,
275.                         nameDataSetExcel)
276.         xl_workbook = xlrd.open_workbook(fileExcel)
277.         sheet_names = xl_workbook.sheet_names()
278.         xl_sheet = xl_workbook.sheet_by_name(sheet_names[0])
279.         xl_sheet = xl_workbook.sheet_by_index(0)
280.
281.         num_cols = xl_sheet.ncols
282.         for row_idx in range(0, xl_sheet.nrows):
283.             for col_idx in range(0, num_cols):
284.                 cell_obj = xl_sheet.cell(row_idx, col_idx)
285.                 stringData = cell_obj.value
286.                 if col_idx == 1:
287.                     stringCaseFolding = PreprocessingCaseFolding(stringData)
288.                     ListStringToken = PreprocessingTokenization(stringCaseFolding)
289.
290.                     ListStringStopWordRemoval = PreprocessingStopWordRemoval(ListStringToken)
291.                     worksheet.write(row_idx, col_idx, stringData)
292.                     if row_idx == 0 and col_idx == num_cols - 1:
293.                         HeaderName = HeaderRow()
294.                         i = 0
295.                         for col_i in range(col_idx + 1, col_idx + len(HeaderName) + 1):
296.                             worksheet.write(row_idx, col_i, HeaderName[i])
297.                             i += 1
298.                         if (row_idx != 0 and col_idx == num_cols - 1):
299.                             i = 0
300.
301.                         partOfSpeechCount = FeatureExtractionPartOfSpeechCount(ListStringTo
ken)
302.                         for col_i in range(col_idx + 1, col_idx + len(HeaderRow()) + 1):
303.                             count = 0;
304.                             if (i == 0):

```

```

302.                                     count = FeatureExtractionWordCount(listStringToken)
303.         elif (i == 1):
304.             count = FeatureExtractionLongWordCount(listStringToken)
305.         elif (i == 2):
306.             count = partOfSpeechCount['noun']
307.         elif (i == 3):
308.             count = partOfSpeechCount['verb']
309.         elif (i == 4):
310.             count = FeatureExtractionCommaCount(stringCaseFolding)
311.         elif (i == 5):
312.
313.             count = FeatureExtractionPunctuationCount(stringCaseFold
314.                                         ing)
315.             elif (i == 6):
316.
317.                 count = FeatureExtractionSentenceCount(stringCaseFolding
318.                                         )
319.             elif (i == 7):
320.                 count = partOfSpeechCount['adjective']
321.             elif (i == 8):
322.                 count = partOfSpeechCount['adverb']
323.             elif (i == 9):
324.
325.                 count = FeatureExtractionLexicalDiversity(listStringToke
326.                                         n)
327.             elif (i == 10):
328.
329.                 count = FeatureExtractionQuotationMark(stringCaseFolding
330.                                         )
331.             elif (i == 11):
332.                 count = FeatureExtractionWordLength(listStringToken)
333.             elif (i == 12):
334.                 count = FeatureExtractionBracketCount(stringCaseFolding)
335.             elif (i == 13):
336.
337.                 count = FeatureExtractionSpellErrorCount(listStringToken
338.                                         )

```

```

329.             elif (i == 14):
330.
331.                 count = FeatureExtractionExclamationMarks(stringCaseFold
332.                                               ing)
333.             elif ( i== 15):
334.
335.                 count = FeatureExtractionForeignWordCount(ListStringToke
336.                                               n)
337.             elif (i == 16):
338.
339.                 count = FeatureExtractionNGram(ListStringStopWordRemoval
340.                                               ,2)
341.             else:
342.
343.                 count = FeatureExtractionNGram(ListStringStopWordRemoval
344.                                               ,3)
345.                 worksheet.write(row_idx, col_i, count)
346.                 i += 1
347.             print (row_idx)
348.         workbook.close()
349.     return
350.
351. """
352. ##### Training, CrossValidate , Model Evaluation #####
353. NAME CONVENTION START WITH TCVME + FUNCTION NUMBER
354. #####
355. """
356. """
357. TCVME 01 --> trainAndMeasureLogisticRegressionCrossValidate(X, Y)
358. function to train a Logistic regression classifier and return kappa and rmse
359. score based on feature X and given target Y measure with 10 Fold Cross
360. Validation
361. """
362.
363. def trainAndMeasureLogisticRegressionCrossValidate(X, Y):
364.     #define model logistic regression jumlah_kelas = 10
365.     logreg = LogisticRegression(C=10)
366.     #measure root mean squared error

```

```

357.
    scores = cross_val_score(logreg, X, Y, cv=10, scoring='neg_mean_squared_error')
    rmse_scores = np.sqrt(-scores)
    rmse = rmse_scores.mean()
    logreg.fit(X, Y)
    predicted = cross_val_predict(logreg, X, Y, cv=10)
    #measure kappa
    kappa = metrics.cohen_kappa_score(Y, predicted, weights='quadratic')
    result = {"kappa":kappa, "rmse":rmse}
    return result
366.
367. """
368. TCVME 02 --> trainAndMeasureLinearRegressionCrossValidate(X, Y)
369. function to train a Linear regression and return kappa and rmse
370. score based on feature X and given target Y measure with 10 Fold Cross
371. Validation
372. """
373. def trainAndMeasureLinearRegressionCrossValidate(X, Y):
374.     #define model multilinear regression
375.     reg = linear_model.LinearRegression()
376.     #measure root mean squared error
377.     scores = cross_val_score(reg, X, Y, cv=10, scoring='neg_mean_squared_error')
378.     rmse_scores = np.sqrt(-scores)
379.     rmse = rmse_scores.mean()
380.     #measure kappa score
381.     reg.fit(X, Y);
382.     predicted = cross_val_predict(reg, X, Y, cv=10)
383.     temp= predicted.round()
384.     #measure kappa
385.     kappa = metrics.cohen_kappa_score(Y, temp, weights='quadratic')
386.     result = {"kappa":kappa, "rmse":rmse}
387.     return result
388. """
389. TCVME 03 --> trainAndMeasureLinearRegressionSingleFeature(X, Y)
390. function to train a Liear regression model and return kappa and rmse

```

```

391.      score based on every single feature in X and given target Y measure with
392.      10 Fold Cross Validation and return List of features sorted by its Accuracy
393.      score
394.      """
395.      def trainAndMeasureLinearRegressionSingleFeature(X, Y):
396.
397.          feature_name =['word_count','long_word_count','noun_count','verb_count','comma_c
398.          ount','punctuation_count','sentence_count','adjective_count','adverb_count','lex
399.          ical_diversity_count','quotaion_mark_count','average_word_length','bracket_count
400.          ','spelling_error_count','exlamation_marks_count','foreign_word_count']
401.
402.          kappaAccuracy = {}
403.
404.          rmseAccuracy = {}
405.
406.          #measure the kappa and rmse score for each feature
407.          for i in range(0,16):
408.
409.              accuracy = trainAndMeasureLinearRegressionCrossValidate(X[:,i:i+1], Y)
410.
411.              kappaAccuracy[feature_name[i]] = accuracy['kappa']
412.
413.              rmseAccuracy[feature_name[i]] = accuracy['rmse']
414.
415.          #sort features based on score
416.
417.          #kappa need to be maximum
418.
419.
420.          kappaAccuracySorted = OrderedDict(sorted(kappaAccuracy.items(), key=
421.          lambda (k,v): (v,k), reverse=True))
422.
423.          #rmse need to be minimum
424.
425.
426.          rmseAccuracySorted = OrderedDict(sorted(rmseAccuracy.items(), key=lambda
427.          (k,v): (v,k)))
428.
429.          return {'kappaList':kappaAccuracySorted, 'rmseList':rmseAccuracySorted }
430.
431.      """
432.
433.      TCVME 03 --> trainAndMeasureLogisticRegressionSingleFeature(X, Y)
434.      function to train a Logistic regression model and return kappa and rmse
435.      score based on every single feature in X and given target Y measure with
436.      10 Fold Cross Validation and return List of features sorted by its Accuracy
437.      score
438.      """
439.
440.      def trainAndMeasureLogisticRegressionSingleFeature(X, Y):

```

```

417.
    feature_name =['word_count', 'long_word_count', 'noun_count', 'verb_count', 'com
        ma_count', 'punctuation_count', 'sentence_count', 'adjective_count', 'adverb_cou
        nt', 'lexical_diversity_count', 'quotaiion_mark_count', 'average_word_length', 'b
        racket_count', 'spelling_error_count', 'exlamation_marks_count', 'foreign_word_
        count']

418.     kappaAccuracy = {}
419.     rmseAccuracy = {}
420.     #measure the kappa and rmse score for each feature
421.     for i in range(0,16):
422.         accuracy = trainAndMeasureLogisticRegressionCrossValidate(X[:,i:i+1], Y)
423.         kappaAccuracy[feature_name[i]] = accuracy['kappa']
424.         rmseAccuracy[feature_name[i]] = accuracy['rmse']
425.     #sort features based on score
426.     #kappa need to be maximum
427.
428.     kappaAccuracySorted = OrderedDict(sorted(kappaAccuracy.iteritems(), key=lambda (k,v): (v,
429.                                                 k), reverse=True))
430.     #rmse need to be minimum
431.
432.
433.     """
434.     TCVME 04 --> generateSortedFeatureXfile(X,Y)
435.     function to generate 4 csv file that containfeatures matriks that already
436.     sorted by its accuray
437.     """
438.     def generateSortedFeatureXfile(X,Y):
439.         sortedColumns = trainAndMeasureLinearRegressionSingleFeature(X,Y)[ 'kappaList' ]
440.         i = 0
441.         #sortedByKappa Linear Reg
442.         for key, value in sortedColumns.iteritems():
443.             if i==0:

```

```

443.                     finalDataFrame = pd.DataFrame({key: data[key]})  

444.             else:  

445.                 tempDataFrame = pd.DataFrame({key: data[key]})  

446.                 finalDataFrame = finalDataFrame.join(tempDataFrame)  

447.             i+=1  

448.         finalDataFrame.to_csv('XSortByKappaLinear.csv', sep=',', encoding='utf-8')  

449.  

450.  

451.         sortedColumns = trainAndMeasureLinearRegressionSingleFeature(X,Y)[‘rmseList’]  

452.         i = 0  

453.         #sortedByRMSE Linear Reg  

454.         for key, value in sortedColumns.iteritems():  

455.             if i==0:  

456.                 finalDataFrame = pd.DataFrame({key: data[key]})  

457.             else:  

458.                 tempDataFrame = pd.DataFrame({key: data[key]})  

459.                 finalDataFrame = finalDataFrame.join(tempDataFrame)  

460.             i+=1  

461.         finalDataFrame.to_csv('XSortByRMSELinear.csv', sep=',', encoding='utf-8')  

462.  

463.  

        sortedColumns = trainAndMeasureLogisticRegressionSingleFeature(X,Y)[‘kappaList’]  

464.         i = 0  

465.         #sortedByKappa Linear Reg  

466.         for key, value in sortedColumns.iteritems():  

467.             if i==0:  

468.                 finalDataFrame = pd.DataFrame({key: data[key]})  

469.             else:  

470.                 tempDataFrame = pd.DataFrame({key: data[key]})  

471.                 finalDataFrame = finalDataFrame.join(tempDataFrame)  

472.             i+=1  

473.         finalDataFrame.to_csv('XSortByKappaLogistic.csv', sep=',', encoding='utf-8')  

474.  

475.  

476.  

        sortedColumns = trainAndMeasureLogisticRegressionSingleFeature(X,Y)[‘rmseList’]  

477.         i = 0

```

```

478.         #sortedByRMSE Linear Reg
479.         for key, value in sortedColumns.iteritems():
480.             if i==0:
481.                 finalDataFrame = pd.DataFrame({key: data[key]})
482.             else:
483.                 tempDataFrame = pd.DataFrame({key: data[key]})
484.                 finalDataFrame = finalDataFrame.join(tempDataFrame)
485.             i+=1
486.         finalDataFrame.to_csv('XSortByRMSELogistic.csv', sep=',', encoding='utf-8')
487.
488.
489. """
490. ##### Features Selection #####
491. NAME CONVENTION START WITH FSEL + FUNCTION NUMBER
492. #####
493. """
494.
495. """
496. FSEL 01 --> forwardFeatureSelectionKappa(X,Y,model)
497. function to do forward features selection using kappa score
498. """
499. def forwardFeatureSelectionKappa(X,Y,model):
500.     #model is Logistic regression
501.     if model == 'Logistic':
502.         featureList = [0]
503.         Xnow = X[:,featureList]
504.
505.         kappaNow = trainAndMeasureLogisticRegressionCrossValidate(Xnow,Y)[ 'kappa
506.             ']
507.         print 'kappa = ', kappaNow
508.         for i in range(1,16):
509.             featureList.append(i)
510.             print i, '--> ', featureList
511.             Xnow = X[:,featureList]
512.
513.             kappaNew = trainAndMeasureLogisticRegressionCrossValidate(Xnow,Y)[ 'kappa
514.             ']

```

```

511.                         #delete features that recently we add cause doesn't increment the
512.                         accuracy
513.                         if(kappaNew <= kappaNow):
514.                             feature List.remove(i)
515.                             print 'hapus kappa = ', kappaNew
516.                         else:
517.                             kappaNow = kappaNew
518.                             print 'kappa = ', kappaNow
519.                         return feature List
520.                         #model is Linear regression
521.                         else:
522.                             feature List = [0]
523.                             Xnow = X[:,feature List]
524.                             kappaNow = trainAndMeasureLinearRegressionCrossValidate(Xnow,Y)[ 'kappa' ]
525.                             print 'kappa = ', kappaNow
526.                             for i in range(1,16):
527.                                 feature List.append(i)
528.                                 print i, '-->', feature List
529.                                 Xnow = X[:,feature List]
530.
531.                                 kappaNew = trainAndMeasureLinearRegressionCrossValidate(Xnow,Y)[ 'kap
532.                                 pa' ]
533.                                 #delete features that recently we add cause doesn't increment the
534.                                 accuracy
535.                                 if(kappaNew <= kappaNow):
536.                                     feature List.remove(i)
537.                                     print 'hapus kappa = ', kappaNew
538.
539.                                     else:
540.                                         kappaNow = kappaNew
541.                                         print 'kappa = ', kappaNow
542.
543.                                         return feature List
544.
545. """
546.     FSEL 02 --> forwardFeatureSelectionKappa(X,Y,model)
547.     function to do forward features selection using RMSE score
548.
549. """
550.
551. def forwardFeatureSelectionRMSE(X,Y,model):

```

```

544.         #model is Logistic regression
545.         if model == 'logistic':
546.             feature List = [0]
547.             Xnow = X[:,feature List]
548.             rmseNow = trainAndMeasureLogisticRegressionCrossValidate(Xnow,Y)[ 'rmse' ]
549.             print 'rmse = ', rmseNow
550.             for i in range(1,16):
551.                 feature List.append(i)
552.                 print i, '--> ', feature List
553.                 Xnow = X[:,feature List]
554.
555.                 rmseNew = trainAndMeasureLogisticRegressionCrossValidate(Xnow,Y)[ 'rmse' ]
556.                 #delete features that recently we add cause doesn't increment the
557.                 #accuracy
558.                 if(rmseNew >= rmseNow):
559.                     feature List.remove(i)
560.                     print 'hapus rmse = ', rmseNew
561.                 else:
562.                     rmseNow = rmseNew
563.                     print 'rmse = ', rmseNow
564.             return feature List
565.         #model is Logistic regression
566.     else:
567.         feature List = [0]
568.         Xnow = X[:,feature List]
569.         rmseNow = trainAndMeasureLinearRegressionCrossValidate(Xnow,Y)[ 'rmse' ]
570.         print 'rmse = ', rmseNow
571.         for i in range(1,16):
572.             feature List.append(i)
573.             print i, '--> ', feature List
574.             Xnow = X[:,feature List]
575.
576.             rmseNew = trainAndMeasureLinearRegressionCrossValidate(Xnow,Y)[ 'rmse' ]
577.
578.             #delete features that recently we add cause doesn't increment the
579.             #accuracy

```

```

575.             if(rmseNew >= rmseNow):
576.                 featureList.remove(i)
577.                 print 'hapus rmse = ', rmseNew
578.             else:
579.                 rmseNow = rmseNew
580.                 print 'rmse = ', rmseNow
581.             return featureList
582.
583. """
584. FSEL 03 --> generateSubset()
585. function to do features subset combinations
586. """
587. def generateSubset():
588.     workbook = xlsxwriter.Workbook("subsetList.xlsx")
589.     worksheet = workbook.add_worksheet()
590.     row = 0; generateSu
591.     col = 0;
592.     listFitur = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
593.     for L in range(0, len(listFitur) + 1):
594.         for subset in itertools.combinations(listFitur, L):
595.             string = map(int, subset);
596.             srings = ' '.join(map(str, string))
597.             worksheet.write(row, col, srings)
598.             row += 1
599.     workbook.close()
600.
601. """
602. FSEL 04 --> wrapperBruteForceFeaturesSelection()
603. Function to do wrapper method for features selection and record the accuracy
604. of each subset in eexcel file(kappa, rmse) using both Linear and Logistic model
605. , to be analyzed further
606. """
607. def wrapperBruteForceFeaturesSelection(X, Y):
608.     X_new = X
609.     LogisticModel = trainAndMeasureLinearRegressionCrossValidate(X, Y)
610.     LinearModel = trainAndMeasureLinearRegressionCrossValidate(X, Y)
611.     workbook = xlsxwriter.Workbook("WrapperLogs.xlsx")

```

```

612.     worksheet = workbook.add_worksheet()
613.     xl_workbook = xlrd.open_workbook("features_subset.xlsx")
614.     sheet_names = xl_workbook.sheet_names()
615.     xl_sheet = xl_workbook.sheet_by_name(sheet_names[0])
616.     xl_sheet = xl_workbook.sheet_by_index(0)
617.
618.     for row_idx in range(64990,65536):
619.         cell_obj = xl_sheet.cell(row_idx, 0)
620.         stringData = cell_obj.value
621.         print row_idx
622.         listString = re.findall(r"\w+", stringData)
623.         subsetCombination = map(int, listString)
624.         X_new = X[:, subsetCombination]
625.         LogisticModel = trainAndMeasureLogisticCrossValidate(X_new,Y)
626.         LinearModel = trainAndMeasureLinearRegressionCrossValidate(X_new,Y)
627.
628.         worksheet.write(row_idx, 0, stringData)
629.         for i in range(1,5):
630.             value =0
631.             #Logistic kappa score
632.             if(i==1):
633.                 value = LogisticModel['kappa']
634.             #Linear kappa score
635.             elif(i==2):
636.                 value = LinearModel['kappa']
637.             #Logistic rmse score
638.             elif(i==3):
639.                 value = LogisticModel['rmse']
640.             #Linear rmse score
641.             elif(i==4):
642.                 value = LinearModel['rmse']
643.             worksheet.write(row_idx,i,value)
644.     Workbook.close()

```