Laporan Praktikum 2

Prediksi Hujan di Denpasar Menggunakan Machine Learning

IF3270 Machine Learning

Rayhan Kinan Muhannad / 13520065

Andhika Arta Aryanto / 13520081

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

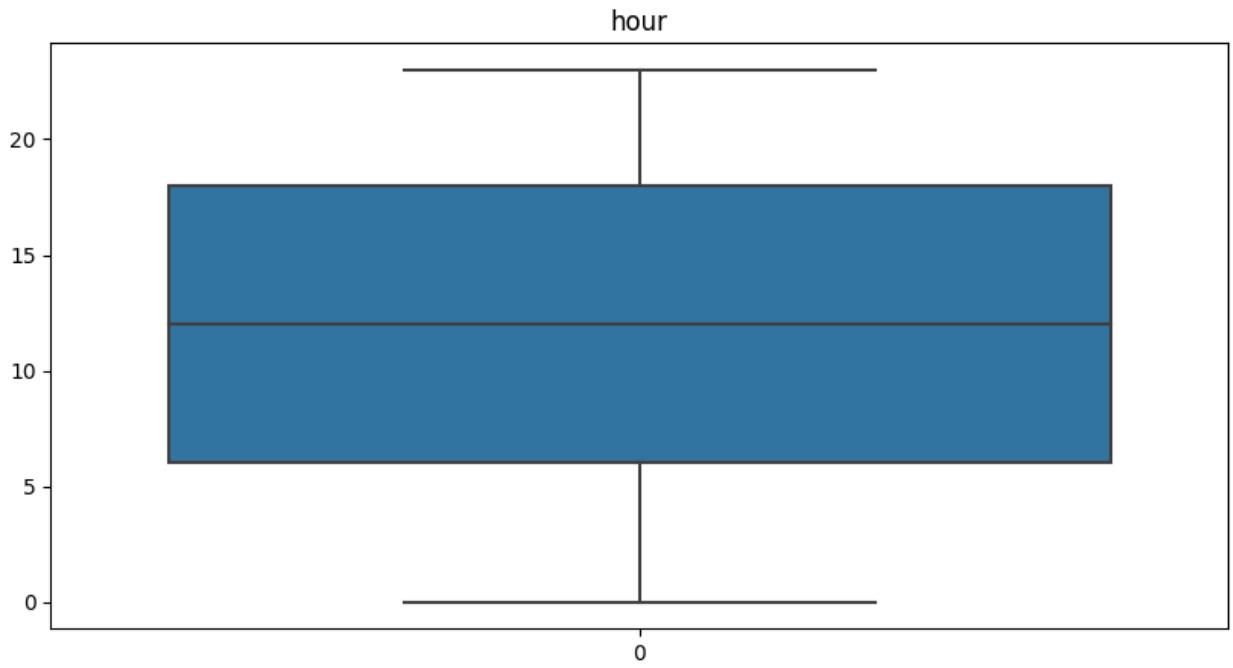
2023

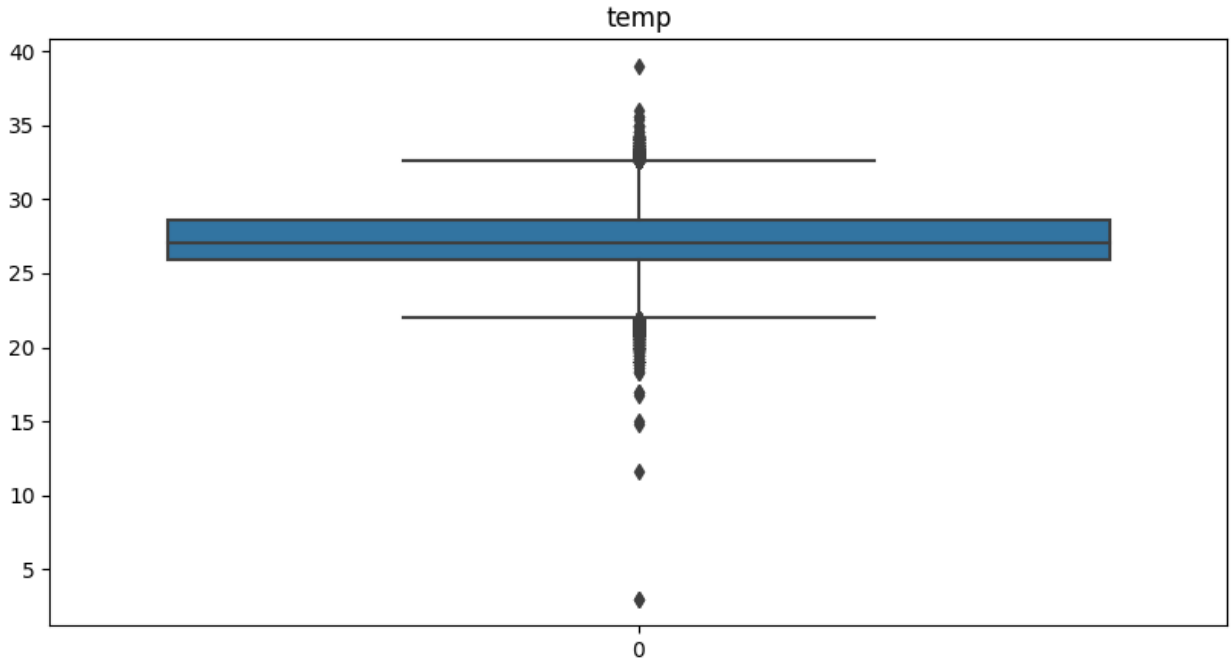
A. Hasil Analisis Data

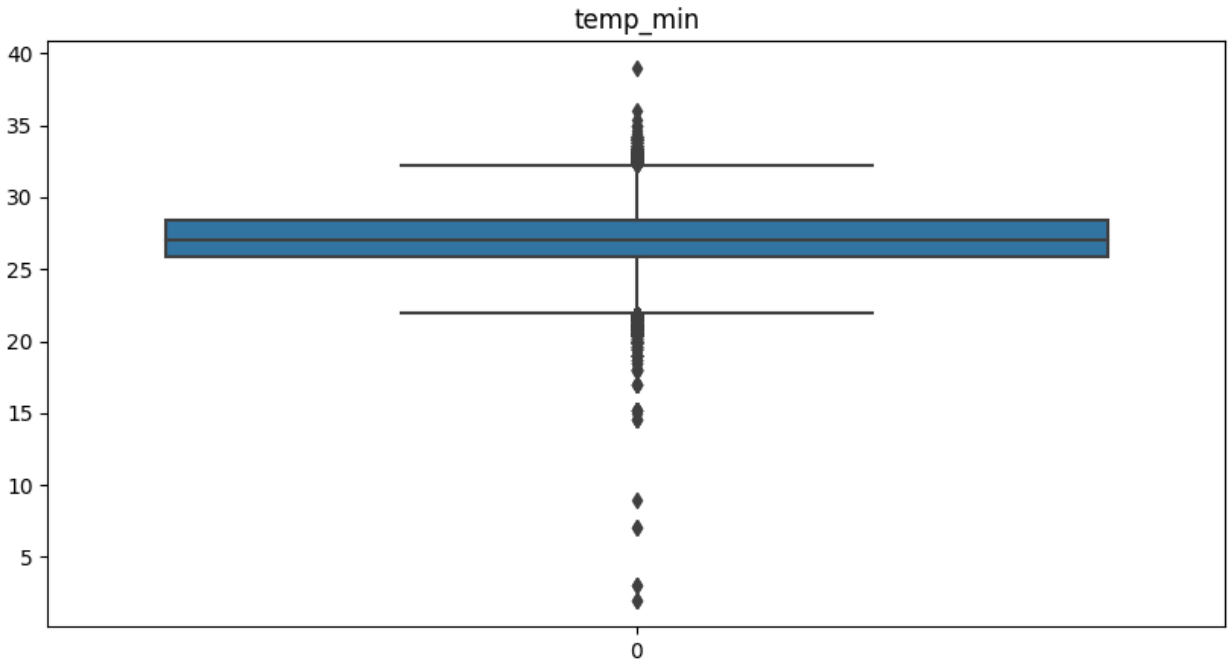
Setelah melakukan analisis pada data cuaca Denpasar, didapatkan beberapa informasi sebagai berikut,

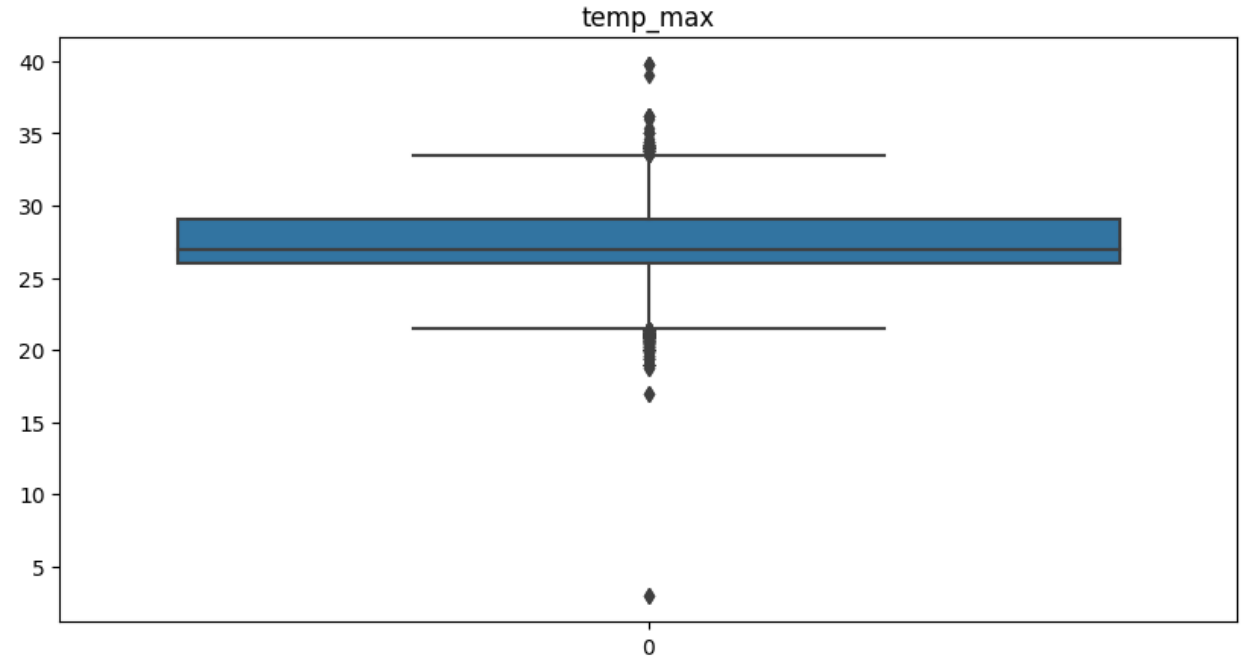
#	Column	Non-Null Count	Dtype
0	hour	264924 non-null	int64
1	temp	264924 non-null	float64
2	temp_min	264924 non-null	float64
3	temp_max	264924 non-null	float64
4	pressure	264924 non-null	float64
5	humidity	264924 non-null	int64
6	wind_speed	264924 non-null	float64
7	wind_deg	264924 non-null	int64
8	raining	264924 non-null	bool

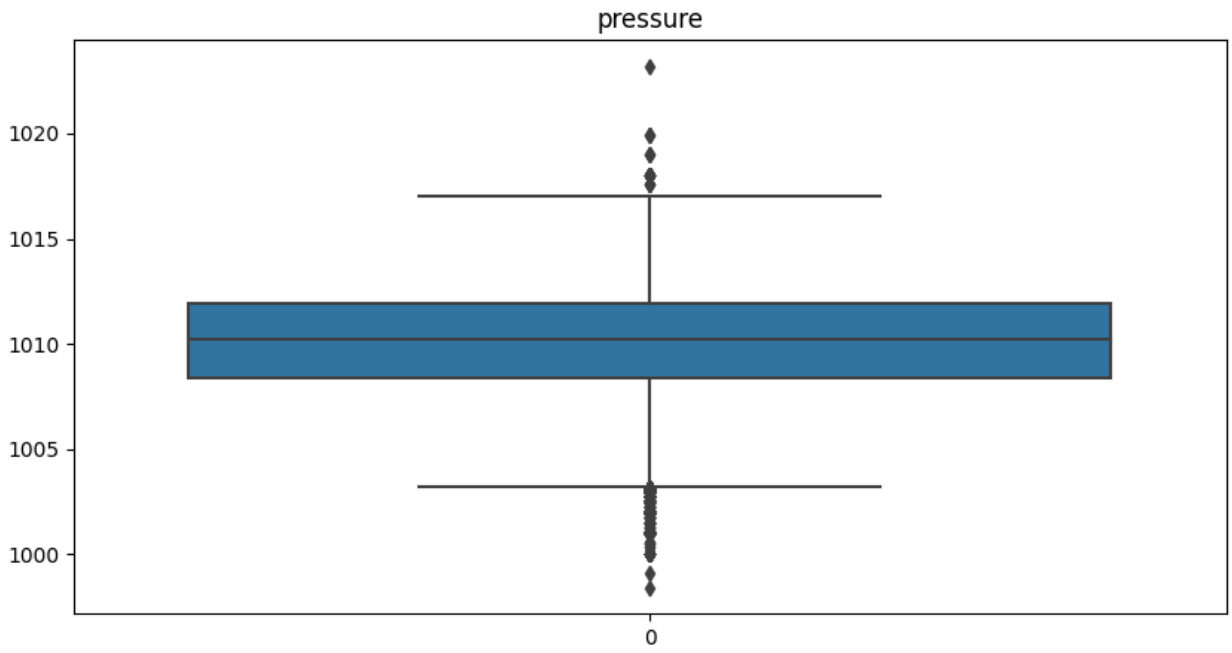
Dataset ini berisi data cuaca dari 1 Januari 1970 sampai 7 Januari 2020. Dataset terdiri dari 9 kolom, 8 atribut fitur dan 1 atribut target. Seluruh atribut fitur bertipe numerik.
Setelah itu dilakukan analisis lebih lanjut untuk mencari nilai duplikat, nilai hilang, *outlier*, dan keseimbangan dari data. Menggunakan fungsi - fungsi terkait, didapat bahwa data memiliki 7253 nilai duplikat dan 0 nilai kosong. Nilai ini bisa dibilang tidak terlalu buruk melihat total baris sebesar 264924. Untuk melihat outlier, dilakukan visualisasi menggunakan boxplot untuk tiap kolom untuk memperlihatkan outlier yang ada. Didapat bahwa terdapat outlier di kolom selain hour dan wind_deg. Berikut boxplot yang dihasilkan :

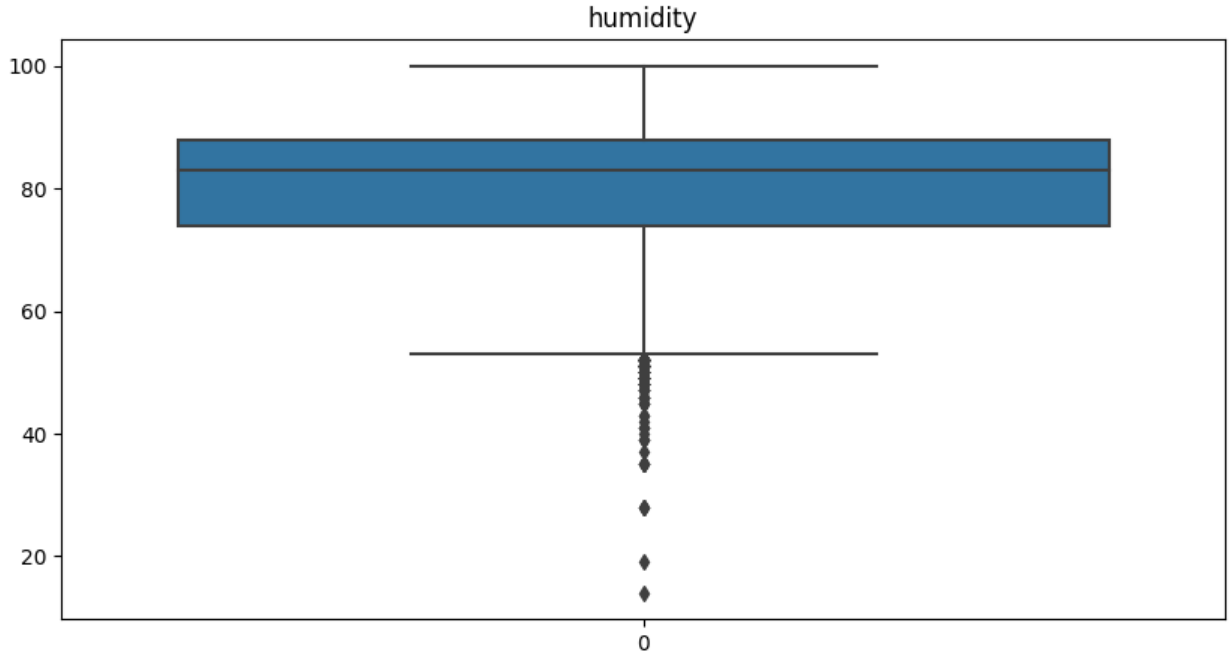

hour
20
15
10
5
0
0

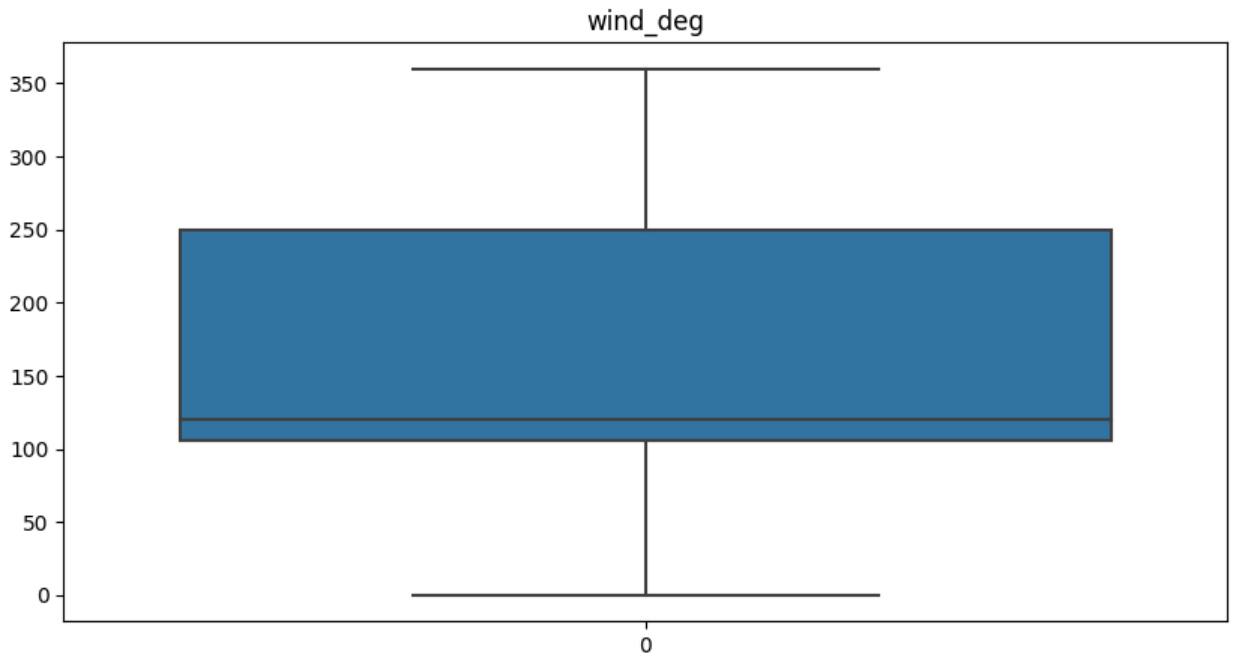

temp
40
35
30
25
20
15
10
5
0

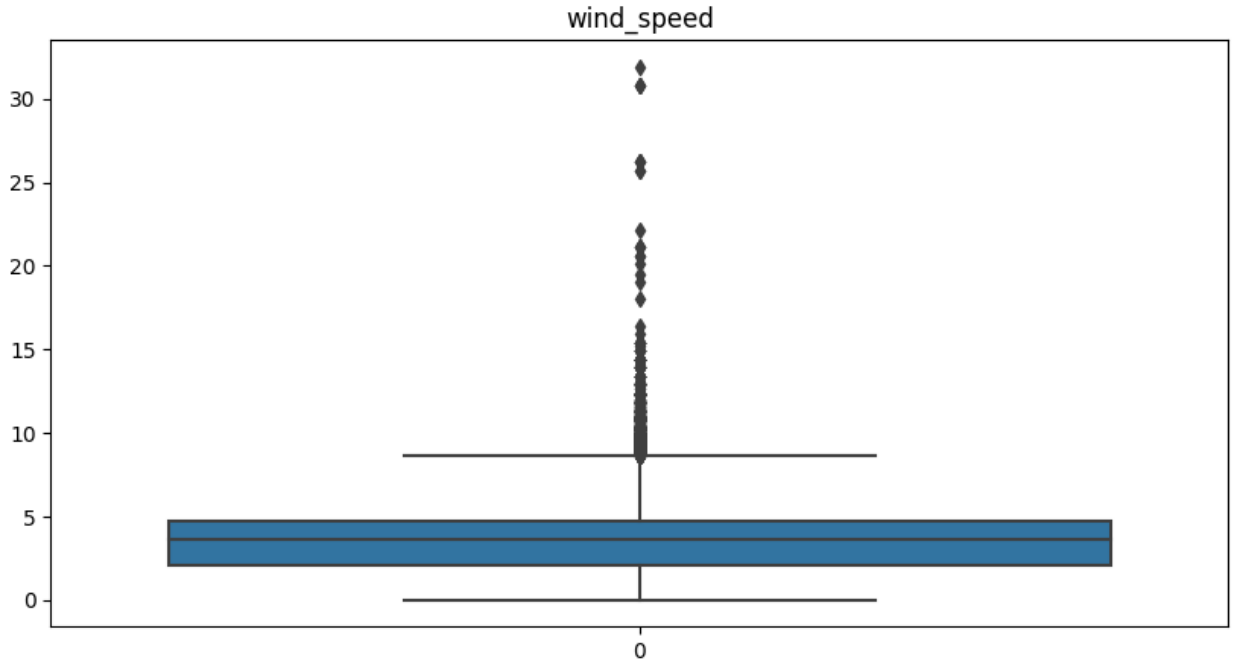

temp_min
40
35
30
25
20
15
10
5
0


temp_max
40
35
30
25
20
15
10
5
0


pressure
1020
1015
1010
1005
1000
0


humidity
100
80
60
40
20
0





Dengan total tiap kolom :

- Fitur hour memiliki 0 outlier
- Fitur temp memiliki 1458 outlier
- Fitur temp_min memiliki 1716 outlier
- Fitur temp_max memiliki 547 outlier
- Fitur pressure memiliki 1067 outlier
- Fitur humidity memiliki 231 outlier

- Fitur wind_speed memiliki 3439 outlier
- Fitur wind_deg memiliki 0 outlier

Outlier dari dataset terbilang tidak terlalu banyak apabila dibandingkan dengan jumlah total dari data.

Terakhir, hal yang dicari adalah keseimbangan dari dataset. Hal ini membandingkan antara data yang bertarget True (hujan) dan False (tidak hujan). Ditemukan :

	Values	Count
0	False	230023
1	True	34901

Dari hasil ini, dapat disimpulkan bahwa data yang dimiliki tidak seimbang karena terlihat bahwa data yang berlabelkan False jauh lebih banyak dari berlabelkan True (hampir 7.5x lipat lebih banyak)

B. Penanganan Data dan Justifikasi

- **Penanganan Data Duplikat**
  Pada dataset ini, akan data yang duplikat akan dihilangkan alias di - drop. Hal ini dilakukan untuk menghilangkan kemungkinan model yang dilatih bias terhadap data yang duplikat tersebut.
- **Penanganan Data Hilang**
  Tidak ada data hilang pada dataset ini, sehingga tidak perlu dilakukan penanganan, beberapa kemungkinan cara yang bisa dilakukan adalah melakukan drop ataupun mengisi nilai hilang dengan beberapa cara misalnya mengisi dengan modus, rata - rata, dan masih banyak cara lainnya.
- **Penanganan Outlier**
  Penanganan outlier dilakukan dengan melakukan drop, hal ini dilakukan karena untuk mengurangi noise, dan membuat model logistic regression menjadi lebih baik karena model ini sensitif terhadap outlier. Alasan lain hal ini dilakukan juga adalah mempercepat kekonvergenan model agar mengurangi kompleksitas dan waktu pada tahap pelatihan model seperti saat grid search dan saat tahap ensemble.
- **Penanganan Imbalanced Dataset**
  Untuk menangani data True yang jauh lebih sedikit , dilakukan metode SMOTE. Kami lebih memilih SMOTE dari undersampling dan oversampling sederhana karena SMOTE dapat meningkatkan akurasi model. Hal ini terbukti saat mencoba melakukan pelatihan baseline model pada ketiga metode ini dan terlihat dari

metrik yang kami pilih (fscore) bahwa SMOTE merupakan metode dengan metrik terbaik

C. Perubahan yang Dilakukan

1. Merubah bagian membuat model menjadi menggunakan 3 model berbeda yaitu DTL, RFC, dan Logistic Regression. Dari ketiga model ini dipilih model terbaik (RFC) dan lalu dilakukan ensemble dengan sebanyak 3 model RFC. Perubahan dilakukan karena model menggunakan SVC membutuhkan waktu terlalu lama saat dilatih (bahkan model normal membutuhkan waktu > 2 jam). Hal ini mungkin disebabkan karena data yang digunakan masih kurang baik sehingga menyebabkan model tidak bisa konvergen

D. Desain Eksperimen

1. Data Preprocessing : Melakukan preprocessing data dengan beberapa tahap. Tahap pertama dilakukan dengan menghilangkan baris yang mengandung *outlier* serta duplikat. Lalu, dilakukan label encoding pada target sehingga mengubah nilai True dan False menjadi 0 dan 1. Hal ini dilakukan untuk memungkinkan pelatihan. Dilanjutkan dengan melakukan *sampling* karena data yang digunakan tidak seimbang. Setelah mencoba beberapa alternatif (oversampling & undersampling sederhana dan SMOTE) ditemukan bahwa SMOTE memberikan nilai metrik yang terbaik
2. Grid Search : Setelah itu, dilakukan grid search pada 3 model yang sudah ditentukan sebelumnya (DTL, RFC, LogReg) untuk menemukan parameter terbaik. Dari sini ditemukan bahwa RFC merupakan model yang memberikan metrik terbaik
3. Model Training : Setelah ditemukan parameter terbaik untuk RFC, dilakukan training dengan menggunakan stacking ensemble menggunakan 3 model RFC dan final estimator dengan menggunakan model LogReg untuk menentukan parameter ensemble yang paling optimum
4. Model Validation : Terakhir, dilakukan validasi dengan dataset test yang sudah dipisahkan sebelumnya

E. Hasil Eksperimen

Dari ekperimen yang dilakukan, didapat beberapa hal sebagai berikut:

- **Untuk Data Sampling, dengan undersampler , oversampler, dan SMOTE didapatkan nilai sebagai berikut**

```
    # Undersampling
    undersampler = RandomUnderSampler(sampling_strategy="majority", random_state=42)
    undersampler_X_train, undersampler_y_train = undersampler.fit_resample(X_train, y_train)

    print(f"Jumlah Sampel: {len(undersampler_X_train)}")

    undersampler_model = LogisticRegression(max_iter=1000, random_state=42)
    undersampler_model.fit(undersampler_X_train, undersampler_y_train)
    undersampler_y_pred = undersampler_model.predict(X_test)

    print(f"Accuracy: {accuracy_score(undersampler_y_pred, y_test)}")
    print(f"F1 Score: {f1_score(undersampler_y_pred, y_test)}")
    print(f"Confusion Matrix:\n{confusion_matrix(undersampler_y_pred, y_test)}")


Jumlah Sampel: 43324
Accuracy: 0.7140993215303706
F1 Score: 0.41552792482252043
Confusion Matrix:
[[30512  1652]
 [12591  5063]]
```

Hasil Pengujian Model dengan Undersampling

```
    # Oversampling
    oversampler = RandomOverSampler(sampling_strategy="minority", random_state=42)
    oversampler_X_train, oversampler_y_train = oversampler.fit_resample(X_train, y_train)

    print(f"Jumlah Sampel: {len(oversampler_X_train)}")

    oversampler_model = LogisticRegression(max_iter=1000, random_state=42)
    oversampler_model.fit(oversampler_X_train, oversampler_y_train)
    oversampler_y_pred = oversampler_model.predict(X_test)

    print(f"Accuracy: {accuracy_score(oversampler_y_pred, y_test)}")
    print(f"F1 Score: {f1_score(oversampler_y_pred, y_test)}")
    print(f"Confusion Matrix:\n{confusion_matrix(oversampler_y_pred, y_test)}")


Jumlah Sampel: 275506
Accuracy: 0.7136577140792485
F1 Score: 0.4145290375538683
Confusion Matrix:
[[30503  1665]
 [12600  5050]]
```

Hasil Pengujian Model dengan Oversampling

```python
# SMOTE
smote_sampler = SMOTE(random_state=42)
smote_X_train, smote_y_train = smote_sampler.fit_resample(X_train, y_train)

print(f"Jumlah Sampel: {len(smote_X_train)}")

smote_model = LogisticRegression(max_iter=1000, random_state=42)
smote_model.fit(smote_X_train, smote_y_train)
smote_y_pred = smote_model.predict(X_test)

print(f"Accuracy: {accuracy_score(smote_y_pred, y_test)}")
print(f"F1 Score: {f1_score(smote_y_pred, y_test)}")
print(f"Confusion Matrix:\n{confusion_matrix(smote_y_pred, y_test)}")
```

```
Jumlah Sampel: 275506
Accuracy: 0.7138383716728893
F1 Score: 0.41497045305318453
Confusion Matrix:
[[30506  1659]
 [12597  5056]]
```

Hasil Pengujian Model dengan SMOTE

Walau perbedaan tidak signifikan, dapat dilihat bahwa sampling dengan SMOTE akan memberikan hasil yang terbaik.

- **Hasil eksperimen Grid Search pada Model LogReg, DTL, dan RFC**

```
# Decision Tree Learning

param_grid = {
    'max_depth': [5, 10, 15],
    'criterion': ['gini', 'entropy'],
}

dtl = DecisionTreeClassifier(random_state=42, splitter='best')
grid_search_dtl = GridSearchCV(estimator=dtl, param_grid=param_grid, n_jobs=-1, verbose=2)
grid_search_dtl.fit(smote_X_train, smote_y_train)
grid_search_y_pred = grid_search_dtl.predict(X_test)

print(f"Parameter: {grid_search_dtl.best_params_}")
print(f"Accuracy: {accuracy_score(grid_search_y_pred, y_test)}")
print(f"F1 Score: {f1_score(grid_search_y_pred, y_test)}")
print(f"Confusion Matrix:\n{confusion_matrix(grid_search_y_pred, y_test)}")


Fitting 5 folds for each of 6 candidates, totalling 30 fits
Parameter: {'criterion': 'gini', 'max_depth': 15}
Accuracy: 0.8381910152956763
F1 Score: 0.4933693671045189
Confusion Matrix:
[[37832  2790]
 [ 5271  3925]]
```

Hasil Pengujian Hyperparam pada DTL

Pada DTL, dilakukan Grid Search pada parameter max_depth yaitu kedalaman pohon dan criterion yaitu kriteria untuk menentukan saat split. Didapat bahwa parameter terbaik adalah menggunakan gini dengan kedalaman 15

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

param_grid = {
    'n_estimators': [10, 20, 30],
    'max_depth': [5, 10, 15, 20],
    'criterion': ['gini', 'entropy'],
}

rf = RandomForestClassifier(random_state=42)
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid, n_jobs=-1, verbose=2)
grid_search_rf.fit(smote_X_train, smote_y_train)
grid_search_y_pred = grid_search_rf.predict(X_test)

print(f"Parameter: {grid_search_rf.best_params_}")
print(f"Accuracy: {accuracy_score(grid_search_y_pred, y_test)}")
print(f"F1 Score: {f1_score(grid_search_y_pred, y_test)}")
print(f"Confusion Matrix:\n{confusion_matrix(grid_search_y_pred, y_test)}")


Fitting 5 folds for each of 24 candidates, totalling 120 fits
Parameter: {'criterion': 'gini', 'max_depth': 20, 'n_estimators': 30}
Accuracy: 0.8672568148058935
F1 Score: 0.5605104007443344
Confusion Matrix:
[[38988  2498]
 [ 4115  4217]]
```

Hasil Pengujian Hyperparam pada RFC

Pada DTL, dilakukan Grid Search pada parameter n_estimator yaitu banyak “tree” yang dibangkitkan dari forest, max_depth yaitu kedalaman masing - masing tree, dan criterion yaitu kriteria saat split. Didapat bahwa penggunaan n_estimator sebanyak 30, max_depth sebanyak 20 dan gini criterion merupakan parameter terbaik

```
# Logistic Regression

param_grid = {
    "solver": ["lbfgs", "liblinear", "newton-cg", "sag", "saga"]
}

log_reg = LogisticRegression(max_iter=1000, random_state=42)
grid_search_log_reg = GridSearchCV(estimator=log_reg, param_grid=param_grid, n_jobs=-1, verbose=2)
grid_search_log_reg.fit(smote_X_train, smote_y_train)
grid_search_y_pred = grid_search_log_reg.predict(X_test)

print(f"Parameter: {grid_search_log_reg.best_params_}")
print(f"Accuracy: {accuracy_score(grid_search_y_pred, y_test)}")
print(f"F1 Score: {f1_score(grid_search_y_pred, y_test)}")
print(f"Confusion Matrix:\n{confusion_matrix(grid_search_y_pred, y_test)}")


Fitting 5 folds for each of 5 candidates, totalling 25 fits
Parameter: {'solver': 'liblinear'}
Accuracy: 0.7139989562005701
F1 Score: 0.4150106749876827
Confusion Matrix:
[[30516  1661]
 [12587  5054]]
```

Hasil Pengujian Hyperparam pada LogReg

Pada LogReg, hanya menggunakan solver, yaitu algoritma yang digunakan saat ada masalah optimasi, didapat bahwa penggunaan liblinear memberikan hasil yang terbaik.
Dari ketiga model diatas, RFC memiliki hasil metrik yang paling baik. Sehingga pada tahap selanjutnya akan digunakan model RFC dengan parameter terbaik yang didapat yaitu 30 n_estimators, 20 max_depth,, dan menggunakan gini.
Setelah itu dilakukan proses training dan validation terhadap model ensemble, didapatkan bahwa tingkat akurasi dan F1 Score dari model jika diujikan menggunakan data testing sudah cukup bagus, yakni 87.25% dan 56.55%. Tingkat akurasi dan F1 Score tersebut terbilang sudah cukup meningkat secara signifikan jika dibandingkan dengan model LogReg yang telah dihitung sebelumnya, yaitu 87,36% dan 22,05%.

Hasil Pengujian Model dengan Data Testing

F. Analisis

Dapat terlihat pada beberapa perhitungan metric pada model bahwa nilai F1 Score yang didapatkan cenderung lebih rendah jika dibandingkan dengan nilai akurasi dari model, dimana umumnya model memiliki nilai akurasi sebesar 80% hingga 90% namun hanya memiliki F1 Score sebesar 20% hingga 50%. Hal tersebut dikarenakan adanya ketidakseimbangan antara jumlah klasifikasi positif dan negatif pada kelas target, dimana tingkat akurasi yang didapatkan pada model cenderung bias terhadap kelas mayoritas (klasifikasi negatif). Oleh karena itu, penulis menggunakan beberapa cara untuk menyeimbangkan jumlah kedua klasifikasi tersebut, diantaranya adalah menggunakan teknik Oversampling, Undersampling, dan juga SMOTE.

Penulis menguji seluruh metode tersebut pada dataset dan menentukan bahwa teknik SMOTE merupakan teknik terbaik untuk menyeimbangkan jumlah klasifikasi target. Teknik SMOTE menyeimbangkan nilai klasifikasi dengan membuat beberapa baris baru pada kolom yang memiliki jumlah klasifikasi yang lebih kecil. Nilai tersebut merupakan hampiran dari beberapa baris yang memiliki kesamaan. Oleh karena itu, teknik SMOTE akan unggul dari Oversampling dikarenakannya adanya bias terhadap perulangan jumlah baris dengan klasifikasi lebih sedikit dan juga unggul dari Undersampling dikarenakan adanya loss of information.

Data hasil pengaplikasian teknik SMOTE tersebut akan digunakan sebagai data training model. Namun, dapat dilihat pula bahwa meskipun terdapat penambahan nilai F1 Score yang signifikan, dapat terlihat bahwa nilai F1 Score tersebut masih jauh dibawah akurasi. Hal tersebut kemungkinan besar diakibatkan oleh data validasi dan data testing yang digunakan juga memiliki jumlah kelas klasifikasi yang tidak seimbang.

Pada dataset yang digunakan, terlihat bahwa jumlah False Negative yang terjadi cenderung lebih tinggi daripada jumlah False Positive. Hal tersebut juga diakibatkan oleh ketidakseimbangan dataset yang cenderung lebih banyak mengandung kelas klasifikasi negatif. Salah satu penyebab mengapa nilai F1 Score cenderung rendah adalah akibat nilai recall yang rendah pula. Nilai recall menghitung jumlah True Positive dibagi dengan jumlah True Positive ditambah dengan jumlah False Negative, dimana jika jumlah False Negative yang tinggi akan mengakibatkan nilai recall yang rendah.
Oleh karena itu, fokus dari eksperimen ini adalah mencari model yang dapat meminimalisir jumlah False Negative dengan mengidentifikasi secara benar nilai yang seharusnya positif namun diklasifikasikan sebagai negatif. Hal ini terbilang sedikit sukar akibat jumlah klasifikasi negatif yang jauh lebih banyak dibandingkan dengan jumlah klasifikasi positif. Model pembelajaran mesin yang baik adalah model yang dapat mengidentifikasi kasus positif secara teliti meskipun menggunakan dataset yang cenderung bias terhadap kasus negatif.

G. Kesimpulan

Pada imbalanced binary dataset yang digunakan di eksperimen ini, dapat disimpulkan bahwa model pembelajaran mesin yang memiliki performa baik adalah model yang memiliki nilai F1 Score yang tinggi. Oleh karena itu, penulis mencoba beberapa model pembelajaran mesin yang secara umum dapat mengklasifikasikan data biner secara baik, seperti Decision Tree Learning, Logistic Regression, dan Random Forest Classification. Selain itu, penulis juga harus melakukan penanganan terhadap imbalanced dataset dengan mencoba beberapa teknik sampling, seperti Oversampling, Undersampling, dan juga SMOTE. Setelah dilakukan beberapa eksperimen, didapatkan bahwa model yang paling optimum dalam mengklasifikasikan dataset ini adalah model Random Forest Classification dengan menggunakan metode sampling SMOTE.
Dari model Random Forest Classification dengan metode sampling SMOTE tersebut, penulis menambahkan model ensemble untuk menggabungkan beberapa Random Forest Classification untuk melakukan agregasi terhadap beberapa model yang dibangun. Penulis menggunakan model Stacking dengan final estimators menggunakan model LogReg sebagai model ensemble utama.
Model ensemble tersebut menghasilkan tingkat akurasi sebesar 87,25% dan nilai F1 Score sebesar 56,55%. Tingkat akurasi tersebut tidak jauh berbeda dengan base model yang digunakan oleh penulis, yaitu 87.36%. Namun, nilai F1 Score yang didapatkan dari model ini jauh lebih baik jika dibandingkan dengan base model dengan nilai 22,06%.

H. Pembagian Tugas

NIM	Nama	Pembagian Tugas
13520065	Rayhan Kinan Muhannad	Grid Search Optimization, Data Training, Laporan
13520081	Andhika Arta Aryanto	Data Preprocessing, Data Training, Laporan