BAB 3
METODOLOGI PENELITIAN

Penelitian ini mengambil data populasi sapi perah di Kecamatan Tutur, Kabupaten Pasuruan, Jawa Timur.

3.1 Metodologi Penelitian

Pada BAB 2 penulis telah menjelaskan mengenai teori dan pendapat tentang siklus estrus sapi perah, penggunaan ANNs untuk memprediksi siklus estrus dengan alat maupun dengan data.

Pada bab ini penulis menjelaskan mengenai alat, bahan, data yang akan dijadikan objek penelitian, waktu beserta lokasi penelitian.

3.1.1 Rencana Kerja

Penelitian ini menggunakan data sapi perah milik Koperasi Peternakan Sapi Perah (KPSP) Setia Kawan Nongkojajar, Kec. Tutur, Kab. Pasuruan. Data mentah sapi perah yang didapatkan penulis akan diolah sesuai

dengan kebutuhan penelitian. Penulis hanya mengambil data sapi perah yang memiliki nomor telinga milik peternak dengan status aktif sebagai anggota KPSP Setia Kawan.

Penelitian ini dilakukan dalam rentang waktu sembilan bulan. Pengumpulan data dilakukan di lokasi objek penelitian dengan cara wawancara dan menyalin pembukuan data digital milik Departemen Kesehatan Hewan KPSP Setia Kawan Nongkojajar. Setelah data diperoleh, penulis melakukan eksperimen di Surabaya. Penggabungan data mentah yang diperoleh sampai menghasilkan dataset yang diperlukan untuk penelitian membutuhkan waktu 4 bulan. Selanjutnya adalah tahapan pengolahan dan analisis data membutuhkan waktu 4 bulan. Untuk penyusunan laporan analisa penelitian membutuhkan waktu 1 bulan.

Rencana kerja waktu pelaksanaan penelitian secara rinci dijelaskan pada Tabel 3.1.

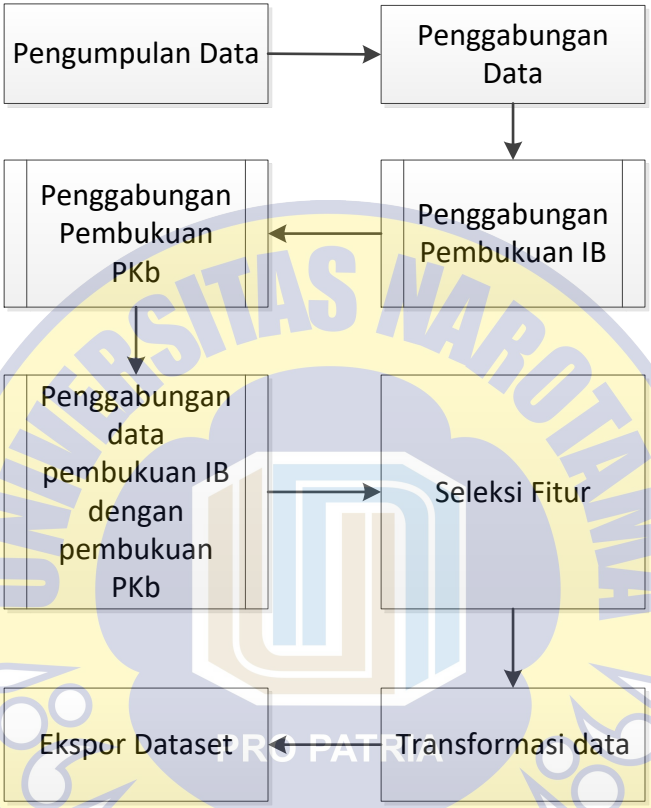
Tabel 3.1. Waktu Pelaksanaan

Proses	Oktober 2018				November 2018				Desember 2018			
	M1	M2	M3	M4	M1	M2	M3	M4	M1	M2	M3	M4
Perancangan proses kerja												
Pengumpulan Data												
Klusterisasi Data												
Data Preprocessing												
Penggabungan Data Pembukuan												
	Januari 2019				Februari 2019				Maret 2019			
	M1	M2	M3	M4	M1	M2	M3	M4	M1	M2	M3	M4
Penggabungan Data Pembukuan												
Seleksi Fitur												
Filter Dataset												
Analisis Dataset												
Analisis dengan Multiple Regresi Logistik												

	April 2019				Mei 2019				Juni 2019			
	M1	M2	M3	M4	M1	M2	M3	M4	M1	M2	M3	M4
Analisis dengan SVC												
Analisis dengan RandomForest												
Analisis dengan ANN												
Pengujian Model dengan Score yang lebih baik												
Merangkum hasil analisis												
Penyusunan Laporan												

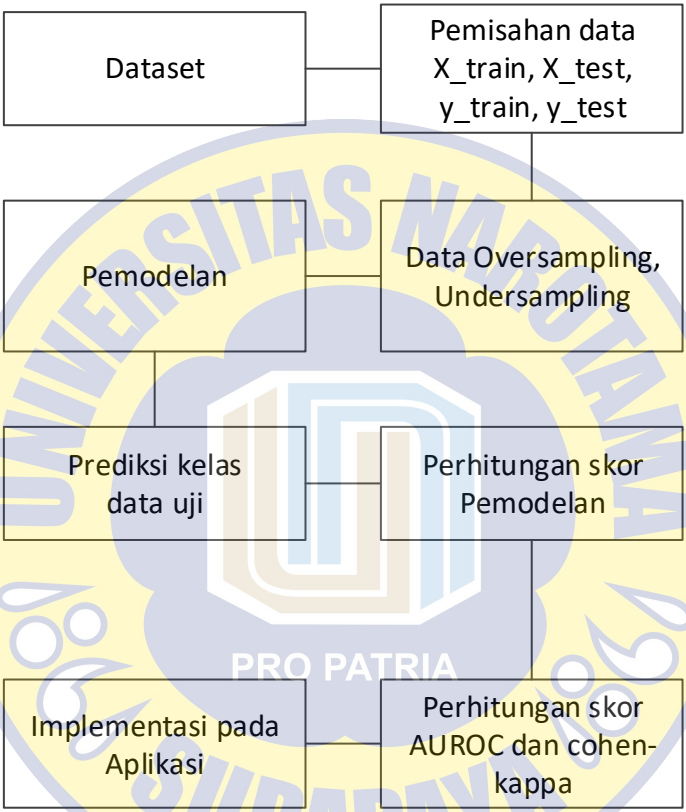
3.1.2 Alur Penelitian

Berdasarkan tahapan-tahapan yang dijelaskan pada sesi 3.1.1, maka penulis membagi penelitian ini menjadi dua bagian, pertama Persiapan Data, kedua Pengolahan / Analisis Data. Penjelasan persiapan data merupakan tahapan awal pengumpulan data sampai akhir *data preprocessing*. Sedangkan pengolahan / analisis data dimulai dari dataset yang telah dihasilkan dari proses sebelumnya sampai pada hasil analisis. Adapun urutan rencana penelitian yang dilakukan penulis seperti yang dijelaskan pada diagram 3.1 berikut.



Gambar 3.1. Diagram persiapan data

Proses pada Gambar 3.1 untuk menghasilkan dataset yang digunakan dalam analisis. Dataset yang digunakan telah memiliki fitur-fitur yang telah diseleksi. Untuk proses selanjutnya, pengolahan dan analisis data dijelaskan pada Gambar 3.2 berikut.



Gambar 3.2. Diagram analisis data

3.1.3 Pengumpulan Data

Data diambil dari Departemen Kesehatan Hewan (Keswan) KPSP Setia Kawan dari tahun 2016 sampai

dengan 2018. Pengolahan dan prediksi siklus estrus, menggunakan data *time series* pada sapi pedet, sapi dara maupun sapi laktasi. Data yang penulis dapatkan berupa data mentah digital berupa data populasi sapi maupun data histori pelaksanaan IB dan dilengkapi dari wawancara dengan petugas keswan. Penulis melakukan sinkronisasi kecocokan kedua data manual dan digital agar tidak mengambil data yang redundan. Data yang kurang lengkap (*missing values*) dilakukan eliminiasi.

Pengumpulan data dilakukan penulis dengan menggabungkan tiga pembukuan rekam data transaksi IB , pemeriksaan hasil susu, dan pemeriksaan kebuntingan (PKb). Pada data rekam transaksi IB, bentuk dari pembukuan seperti ditunjukkan pada Tabel 3.2. Transaksi pencatatan IB terdapat 63.395 rekam data. Untuk pembukuan PKb ditunjukkan pada Tabel 3.3. Transaksi Pemeriksaan Kebuntingan (PKb) sejumlah 15.932 data. Sedangkan pembukuan produksi hasil susu sapi ditunjukkan pada Tabel 3.3. Transaksi Pemeriksaan Kebuntingan (PKb) sejumlah 9.413 data.

Setiap tabel transaksi IB dan PKb memiliki kolom nomor sapi, nama peternak, lokasi, *semen code* dan batch. Kolom-kolom tersebut yang dijadikan penulis sebagai penghubung ketika dilakuan penggabungan data (*joining*) untuk menghindari data yang redundant. Dan setiap kolom pemeriksaan hasil susu dan perilaku sapi memiliki kolom nomor sapi, nama peternak, dan lokasi yang dilakukan penggabungan dari hasil penggabungan pembukuan IB dan Pkb.

3.1.4 Seleksi Fitur

Untuk menangani keberagaman data yang diperoleh, penulis melakukan seleksi fitur. Sapi perah yang dijadikan objek penelitian bisa dimulai pada fase pedet (anakan), dara (remaja) maupun induk (dewasa). Sapi yang pernah melahirkan merupakan sapi induk dewasa laktasi. Berdasarkan data pembukuan rekam data, penulis menyeleksi variabel yang digunakan berdasarkan waktu pencatatan IB dan rentang waktu antar IB. Penulis melakukan eliminasi pada kolom variabel yang tidak memiliki pengaruh terhadap variabel respon yang diharapkan.

Tabel 3.2. Transaksi pencatatan IB

#	nomor_telinga	IB				sem_code	batch	ib_sebelumnya			nm_peternak	alamat
		ib1	ib2	ib3	ib4			ib1	ib2	ib3		
1	153/XII/AK		1/1/2010			30780	HH 19	14/12/2010			SIANAH	MESAGI
2	001/I/AL	2/1/2010				30780	HH 19				MULIONO	WON.BARAT
3	082/X/AG			2/1/2010		30780	HH 19		8/11/2010		YUNIANTO	NGADIPURO
...	…	…	…	…	…	…	…	…	…	…	…	…
63393	202/VI/AP		23/12/2015			307100	AM.103	30/11/2015			SUKIRNO	TEMPURAN
63394	462/XII/AQ	24/12/2015				307100	AM.103				TASIM	TEMPURAN
63395	327/XII/AJ		25/12/2015			307100	AM.103	22/10/2015			NASRIB	TEMPURAN

Tabel 3.3. Transaksi Pemeriksaan Kebuntingan (PKb)

#	num_reg	nm_peternak	alamat	ib_tgl	pkb_tgl	sem_code	batch	ib_ke	ib_hasil
1	248/VIII/AJ	SANAWI	DODOGAN	26/08/2010	26/11/2010	30152	HH032	2	0
2	124/V/AB	SANATI	DODOGAN	26/08/2010	26/11/2010	30152	HH032	1	1
3	054/II/AJ	MUKSIN	JELAG	26/08/2010	26/11/2010	30152	HH032	2	1
...	...	...	...	...	...				
15930	386/IX/AQ	P.SOLIHIN	TEMPURAN	25/09/2015	26/12/2015	31110	N.17	1	1
15931	384/IX/AQ	PANDI	TEMPURAN	25/09/2015	26/12/2015	31110	N.17	1	0
15932	387/IX/AQ	NAWARI	TEMPURAN	25/09/2015	26/12/2015	31110	N.17	1	1

Tabel 3.4. Transaksi Produksi Hasil Susu

#	num_reg	nm_peternak	prod_susu	perilaku	laktasi	alamat
1	/VIII/AJ	ABAS	6	4	3	ANDONG
2	44/VII/AL	ABDOLO	4	3	2	DEMPOK
3	/VIII/AJ	ABDUL ABAS	1	2	4	ANDONG
...	...	...	...	...		...
9411	223/X/AK	YUNUS	6	1	4	CURAH BUNTUNG
9412	129/V/AK	YUSUF	4	2	1	TANJUNG WONO
9413	123/III/AI	YUSWANTO	8	3	5	KUNTUL

3.1.5 Transformasi Data

Berdasarkan data pembukuan manual, penulis melakukan transformasi menjadi tipe data yang dapat dilakukan analisis menggunakan metode yang digunakan. Transformasi data ini terutama berlaku pada data yang memiliki tipe Days untuk diubah dalam bentuk integer melalui perhitungan rentang waktu. Transformasi data juga diterapkan pada variabel output berupa kondisi sapi bunting menggunakan nilai klasifikasi 1 dan 0. Untuk penggabungan dan seleksi kolom, penulis menggunakan tools Knime[13].

Setelah data transaksi IB, PKb dan produksi susu digabung diperoleh 9.413 rekam data dengan kolom-kolom :

‘**num_reg**’, ID sapi

‘**ib_combine**’, hasil pencocokan tanggal IB

‘**pkb_tgl**’, tanggal pemeriksaan kebuntingan

‘**durasi_pkb_ke_ib**’, jarak antara pemeriksaan kebuntingan dengan tanggal IB terakhir

‘**ib_ke**’, transaksi IB terakhir yang ke-sekian

‘**jarak_ib_sebelumnya**’, rentang waktu antar IB

‘**prod_susu**’, *grade /* kualitas produksi hasil susu

‘**perilaku**’, aktivitas fisik tanda-tanda estrus

‘**laktasi**’, status laktasi / kelahiran ke sekian dari induk sapi

‘**ib_hasil**’, hasil dari pemeriksaan kebuntingan

‘**status**’, hasil dari pemeriksaan kebuntingan dalam label

Dari 9.413 rekam data hasil transformasi, masih terdapat baris data yang memiliki nilai nihil (*missing values*) yang dapat mempengaruhi dari karakteristik dataset, sehingga penulis melakukan eliminiasi. Penulis juga melakukan eliminasi terhadap data yang tidak konsisten atau indikasi dari kesalahan entry data termasuk juga data yang *redundant*.

Untuk variable ‘ib_ke’ memiliki nilai peringkat 1 sampai dengan 4. Nilai ib_ke = 1 akan dijadikan baseline pengecualian untuk menghindari multikoliniearitas, sehingga variable ‘ib_ke’ memiliki jenis nilai 2, 3 atau 4.

Tabel 3.5. Dataset

#	num_reg	Prod_susu	perilaku	laktasi	ib_ke	jarak_ib_sebelumnya	ib_hasil
1	/I/AJ	7	1	5	2	20	1
2	/I/AN	7	5	3	2	70	0
3	/I/AO	9	2	1	2	31	0
...	...	...			...	...	...
1801	99/V/AL	8	5	1	2	75	0
1802	99/VI/AL	1	1	5	2	75	0
1803	99/VII/AN	6	4	3	2	79	1

3.1.6 Dataset

Dari data preprocessing, diperoleh dataset final sebanyak 1.803 rekam baris data. Dataset ini memiliki jumlah 5 variabel dengan 1 variabel yang menunjukan sebagai variabel output dalam bentuk Integer (ib_hasil) dan String yaitu (status). Penjelasan Tabel 3.5. DatasetTabel 3.5 adalah sebagai berikut. Variabel 'num_reg' merupakan ID sapi dari peternak. Variabel 'num_reg' sebelumnya memiliki atribut nama peternak beserta lokasi wilayah untuk memastikan bahwa data tidak redundant. Namun pada dataset final, penulis hanya mengambil 'num_reg' sebagai variabel ID sapi.

Variabel 'ib_ke' merupakan hasil *joining* rekam data transaksi IB, dari rekam data IB yang gagal sampai IB yang berhasil sapi bunting. Setiap rekam IB yang gagal sapi bunting, akan dilakukan IB lagi dengan pencatatan akumulasi dari IB sebelumnya. Sebagai contoh, Sapi A dilakukan IB pertama kali (IB ke 1). Namun pada waktu yang seharusnya masuk pada fase *metestrus* atau fase setelah birahi, sapi A masih menunjukkan gejala birahi. Dapat diasumsikan proses IB ke 1 gagal sapi bunting atau

gagal dibuahi. Maka sapi A dilakukan IB lagi dengan rekam data IB kedua (IB ke 2). Begitu seterusnya sampai maksimum IB ke 4.

Variabel 'jarak_ib_sebelumnya' merupakan perhitungan rentang waktu dari jarak masing-masing pencatatan rekam IB. Untuk mencari nilai variabel ini, penulis menggunakan ID sapi 'num_reg' milik peternak pada lokasi yang sama agar tidak terjadi *missed-classification* atau dalam artian perhitungan 'jarak_ib_sebelumnya' pada sapi yang sama. Selanjutnya variabel 'ib_ke' dan 'jarak_ib_sebelumnya' berperan sebagai fitur variabel bebas.

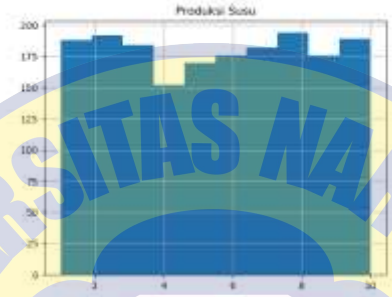
Variabel 'ib_hasil' merupakan variabel *output* / variabel respon dengan nilai 1 atau 0 sebagai representasi 'Diharapkan Estrus' atau 'Terlambat Estrus'. Nilai variabel 'ib_hasil' diperoleh pada rekam pembukuan dengan status kebuntingan sapi. Apabila status kebuntingan sapi setelah proses PKb menunjukan hasil positif, maka penulis mentransformasi variabel 'ib_hasil' dengan nilai 1, begitu sebaliknya. Impelementasi dalam kode Python sebagai berikut.

```
In []:
import pandas as pd
df = pd.read_csv(SeleksiOK.csv')
df = pd.DataFrame(df,
    columns=['ib_ke','jarak_ib_sebelumnya','ib_hasil'])
print(df.head(5))
Out []:
   prod  perilaku  ib_ke  jarak_ib  laktasi  ib_hasil
0     8         2      2        45        1         1
1     3         4      2        24        3         1
2     9         1      2        28        5         1
3     1         3      2        39        4         1
4    10         1      2        21        5         1
```

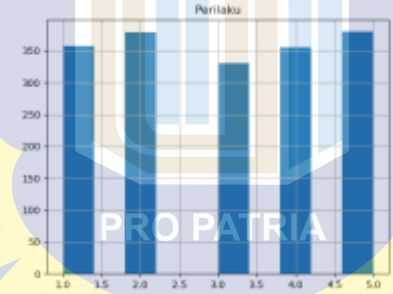
Visualisasi dataset dalam implementasi kode Python sebagai berikut.

```
In []:
import matplotlib.pyplot as plt
df.prod.hist()
plt.title(Produksi Susu')
plt.show()
df.perilaku.hist()
plt.title(Perilaku')
plt.show()
df.ib_hasil.hist()
plt.title('IB ke-Sekian')
plt.show()
df.jarak_ib_sebelumnya.hist()
plt.title('Jarak IB Sebelumnya')
plt.show()
df.laktasi.hist()
plt.title(Laktasi')
plt.show()
df.ib_hasil.hist()
plt.title('Hasil IB')
plt.show()
```

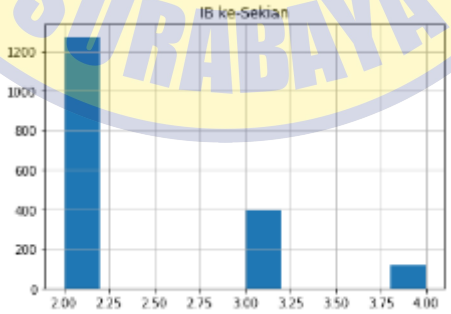
Visualisasi histogram dari masing-masing variabel ditunjukan pada Gambar 3:3.



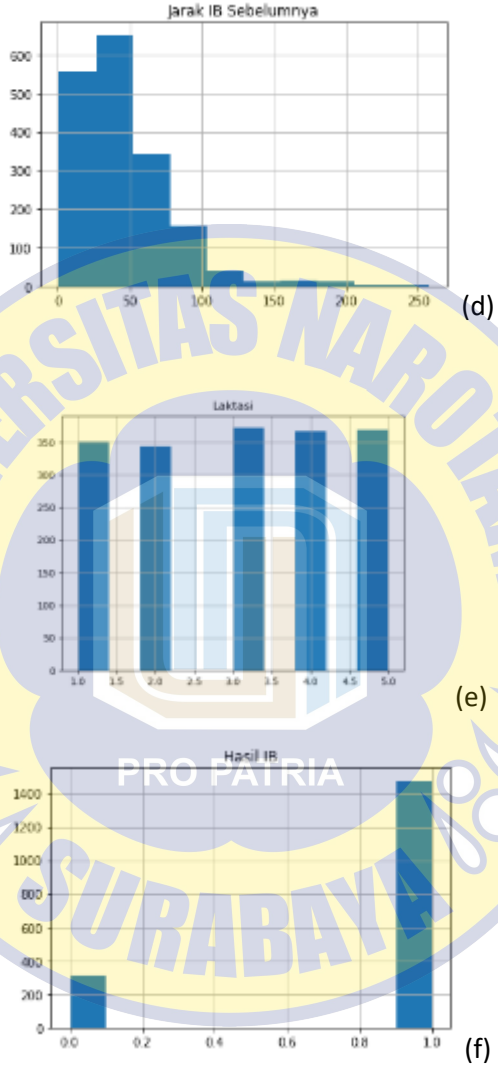
(a)



(b)



(c)



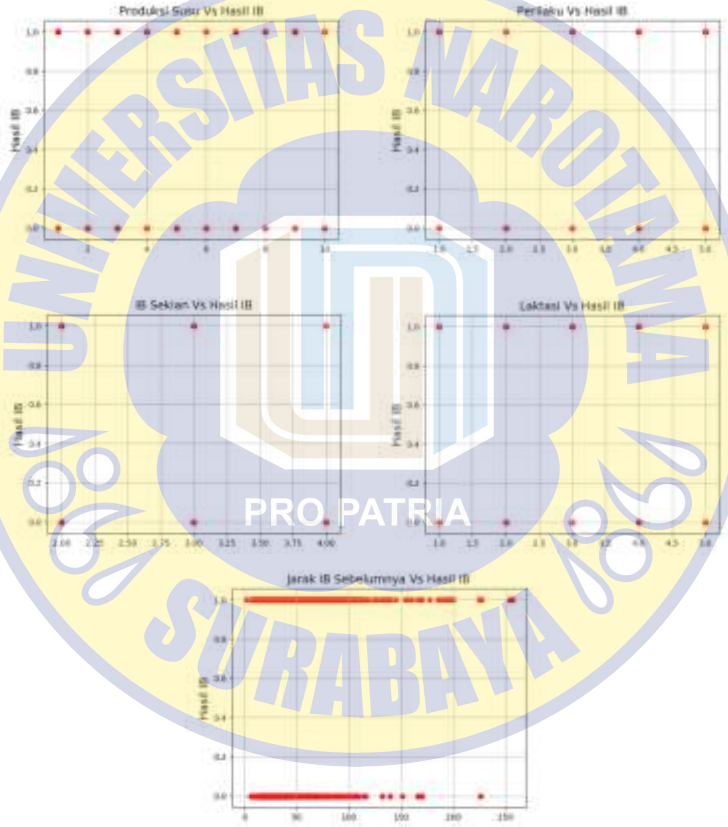
Gambar 3.3. Histogram (a) Variabel prod_susu, (b) variabel perilaku, (c) variabel ib_hasil, , (d) variabel ib_hasil, (e) variabel laktasi, (f) variabel hasil_ib

Karena hasil output berupa data dikotomi kategorikal dengan variabel bebas lebih dari satu, maka perlu dilakukan uji liniearitas dari masing-masing variabel bebas terhadap variabel respon. Implementasi pada kode Python sebagai berikut.

```
In []:
plt.scatter(df['prod_susu'], df['ib_hasil'], color='red')
plt.title('Produksi Susu Vs Hasil IB', fontsize=14)
plt.xlabel('Produksi Susu, fontsize=14)
plt.ylabel('Hasil IB', fontsize=14)
plt.grid(True)
plt.show()

plt.scatter(df['perilaku'], df['ib_hasil'], color='red')
plt.title('Perilaku Vs Hasil IB', fontsize=14)
plt.xlabel('Perilaku, fontsize=14)
plt.ylabel('Hasil IB', fontsize=14)
plt.grid(True)
plt.show()

plt.scatter(df['ib_ke'], df['ib_hasil'], color='red')
plt.title('IB Sekian Vs Hasil IB', fontsize=14)
plt.xlabel('IB Sekian, fontsize=14)
plt.ylabel('Hasil IB', fontsize=14)
plt.grid(True)
plt.show()

plt.scatter(df['jarak_ib_sebelumnya'], df['ib_hasil'],
color='red')
plt.title(Jarak IB vs Hasil IB', fontsize=14)
plt.xlabel(Jarak IB', fontsize=14)
plt.ylabel('Hasil IB', fontsize=14)
plt.grid(True)
plt.show()

plt.scatter(df['laktasi'], df['ib_hasil'], color='red')
plt.title('Laktasi Vs Hasil IB', fontsize=14)
```

```
plt.xlabel('Laktasi, fontsize=14)
plt.ylabel('Hasil IB', fontsize=14)
plt.grid(True)
plt.show()
```

Hasil uji linearitas variabel bebas dengan variabel respon ditunjukan pada Gambar 3.4.



Gambar 3.4. Diagram Scatter variabel bebas dengan variabel respon

3.1.7 Penanganan Imbalanced Data

Dari 1.803 baris rekam data yang digunakan, diperoleh variabel respon hasil 'ib_hasil' dengan nilai 1 sejumlah 1.493 dan nilai 0 sejumlah 310 dengan presentase rasio 83% : 17%. Untuk menghindari skor pemodelan yang anomali maka penulis melakukan *resampling* dataset untuk medapatkan rasio yang seimbang antara nilai 1 dan 0.

Proses *resampling* ini memiliki dua metode, *oversampling* dan *undersampling*. *Oversampling* merupakan metode menaikan rasio data minor agar seimbang data mayor. Sedangkan *undersampling* merupakan menurunkan rasio data mayor agar seimbang dengan data minor. Penulis melakukan resampling dataset menggunakan *library scikit-learn*[14] metode *resample*. Implementasi pada kode Python sebagai berikut.

```
In []:
from sklearn.utils import resample
df_majority = df[df.ib_hasil==1]
df_minority = df[df.ib_hasil==0]
# Oversampling minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,
                                 n_samples=1493,
                                 random_state=1)
```

```
# Kombinasi
df = pd.concat([df_majority, df_minority_upsampled])

# Undersampling majority class
df_majority_downsampled = resample(df_majority,
                                   replace=False,
                                   n_samples=310,
                                   random_state=1)
# Kombinasi
df = pd.concat([df_majority_downsampled, df_minority])
```

3.1.8 Data Latih dan Data Uji

Sebelum melakukan pemodelan pada dataset yang digunakan, penulis memisah data latih (*training)* dengan data uji (*testing)*. Dalam beberapa percobaan, penulis menggunakan rasio data latih dengan data uji 80:20. Penulis menggunakan library *scikit-learn* paket *model_selection* metode *train_test_split*. Implementasi pada kode Python sebagai berikut.

```
In []:
from sklearn.model_selection import train_test_split
X = df.drop('ib_hasil', axis=1)
y = df['ib_hasil']
X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.2, random_state=1)
```

3.2 Metode Penelitian

Berdasarkan bentuk dataset yang digunakan yaitu memiliki variabel respon berupa data dikotomi 1 dan 0 maka pemodelan yang digunakan menggunakan pendekatan algoritma probabilitas untuk menagani data yang telah diklasifikasi atau biasa disebut *supervised learning algorithm*. Sebelum melakukan pemodelan dengan ANN, penulis melakukan eksperimen menggunakan algoritma statistik probabilitas untuk mendapat nilai perbandingan skor dari setiap pemodelan. Sesi ini membahas langkah-langkah pemodelan yang dilakuan penulis.

Penulis melakukan analisis uji dari masing-masing pemodelan dengan menghitung skor prediksi model, skor *Area Under the Receiver Operating Characteristics* (AUROC), dan skor *cohen-kappa*.

3.2.1 Analisis Regresi Logistik

Penulis menggunakan pemodelan metode regresi logistik berganda multivariate dengan pendekatan distribusi Bernoulli. Variabel respon yang dijadikan sebagai output adalah ib_hasil dengan nilai 1 atau 0. Pemodelan dilakukan

sebanyak 3 kali uji coba, pertama menggunakan data asli dengan perbandingan frekuensi variabel respon bernilai 1:0 sejumlah 1.493:310, menggunakan *oversampling* dengan perbandingan 1.493:1.493 dan *undersampling* dengan perbandingan 310:310.

Implementasi pada kode Python menggunakan library *scikit-learn* paket *linear-model* metode *LogisticRegression*.

```
In []:
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

clf = LogisticRegression(solver='lbfgs').fit(X_train, y_train)
y_pred = clf.predict(X_test)

accuracy_score(y_test, y_pred)
```


3.2.2 Analisis Neural Networks

Penulis menggunakan library *scikit-learn* paket *neural_networks* modul *MLPClassifier*. Pemodelan ANN menggunakan algoritma *backpropagation* dengan fungsi aktivasi *tanh*. Penulis melakukan uji coba pada dataset asli dengan rasio 1.493:310, dataset *oversampling* dengan rasio 1.493:1.493 dan dataset *undersampling* dengan rasio

310:310. Uji coba dilakukan menggunakan parameter 3 hidden layer dan 20 neuron dengan iterasi maksimum sebanyak 200 kali. Uji coba pertama menggunakan komposisi rasio data asli. Implementasi pada kode Python sebagai berikut.

```
In []:
from sklearn.neural_networks import MLPClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import cohen_kappa_score, roc_auc_score

X_train = MinMaxScaler().fit_transform(X_train)
X_test = MinMaxScaler().fit_transform(X_test)

mlp = MLPClassifier(hidden_layer_sizes=(6,8,6),
                    max_iter=200,
                    activation='tanh',
                    random_state=1,
                    solver='lbfgs',
                    warm_start=False) #3 layer, 20 neurons, 200
iterations

mlp.fit(X_train, y_train) #fit model
print(mlp.fit(X_train, y_train))
y_pred = mlp.predict(X_test)

print("\n",np.unique(y_pred))
print(mlp.score(X_test, y_test)) #skor mlp
print(cohen_kappa_score(y_test, y_pred)) #skor cohen kappa
print(roc_auc_score(y_test, y_pred)) #skor ROC AUC
Out []:
MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto',
beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=6,8,6, learning_rate='constant',
              learning_rate_init=0.001, max_iter=200, momentum=0.9
,n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=1, shuffle=True, solver='lbfgs', tol=0.
0001,validation_fraction=0.1, verbose=10, warm_start=False)

[0 1]
```

```
0.8005540166204986 #skor mlp
0.06698255438294198 #skor cohen kappa
0.5221889269406393 #skor ROC AUC
```

Untuk uji coba kedua, akan menggunakan dataset *oversampling*. Hasil luaran pada kode Python sebagai berikut.

```
Out []:
MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto',
beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hid
den_layer_sizes=6,8,6, learning_rate='constant', learning_rate_ini
t=0.001, max_iter=200, momentum=0.9, n_iter_no_change=10, nesterov
s_momentum=True, power_t=0.5, random_state=1, shuffle=True, solver
='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=10, warm_st
art=False)

[0 1]
0.5936454849498328 #skor mlp
0.19260131793885915 #skor cohen kappa
0.5969871967051661 #skor ROC AUC
```

Uji coba kedua, akan menggunakan dataset *undersampling*. Hasil luaran pada kode Python sebagai berikut.

```
Out []:
MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto',
beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hid
den_layer_sizes=6,8,6, learning_rate='constant', learning_rate_ini
t=0.001, max_iter=200, momentum=0.9, n_iter_no_change=10, nesterov
s_momentum=True, power_t=0.5, random_state=1, shuffle=True, solver
='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=10, warm_st
art=False)
```

```
[0 1]
0.45161290322580644 #skor mlp
-0.09449636552440288 #skor cohen kappa
0.4526041666666667 #skor ROC AUC
```

3.2.3 Hidden Layer dan Neuron

Parameter *hidden layer* dan *neuron* dapat disesuaikan untuk mendapatkan hasil skor akurasi yang bervariasi. Penulis melakukan uji coba dengan beberapa macam jumlah *hidden layer* dan *neuron*. Implementasi pada kode Python sebagai berikut.

```
mlp = MLPClassifier(hidden_layer_sizes=(6,8,6),
                    max_iter=200,
                    activation='tanh',
                    random_state=1,
                    solver='lbfgs',
                    warm_start=False
                    )
```

Nilai argumen hidden_layer_size menunjukan jumlah *neuron* yang digunakan, sedangkan jumlah parameter menunjukan banyaknya *hidden layer*. Pada contoh kode Python diartikan bahwa *hidden layer* pertama menggunakan dua neuron, *hidden layer* kedua menggunakan empat neuron dan *hidden layer* ketiga menggunakan enam neuron. Iterasi yang dilakukan sebanyak dua ratus, library MLPClassifier akan berhenti

melakukan iterasi jika nilai toleransi lebih kecil dari 0,00001.

Pemodelan menggunakan ANN melibatkan kalkulasi bobot antar lapisan *neuron*. Pembobotan yang dihasilkan *method* MLPClassifier dapat dilihat dengan memanggil argument *coefs_* dan nilai bias dengan argumen *intercepts_*. Jumlah bobot yang digunakan setiap *neuron* mengikuti jumlah *neuron* pada lapisan selanjutnya dengan dimensi sebanyak jumlah layer-1. Pembobotan antar layer dihasilkan secara random dari modul *MLPClassifier*.

3.2.4 Prediksi Data Uji

Data prediksi dihasilkan dari variabel bebas data latih (X_test) menggunakan pemodelan. Kesalahan dalam penggunaan parameter data prediksi akan menghasilkan galat. Dimensi data prediksi harus sama dengan dimensi data uji variabel respon. Implementasi pada kode Python sebagai berikut.

```
y_pred = mlp.predict(X_test)
```

3.2.5 Penilaian Model

Untuk menguji akurasi pemodelan yang digunakan, penulis menggunakan tiga penilaian yaitu skor akurasi, skor

AUROC dan skor *cohen-kappa*. Penilaian skor akurasi menggunakan data uji variabel respon dengan komposisi 20% dari populasi terhadap data hasil prediksi. Dari masing-masing uji coba pertama, kedua, dan ketiga memiliki dimensi data uji yang berbeda. Penilaian skor akurasi menunjukan presentase dari pemodelan berdasarkan data populasi. Pada *library MLPClassifier*, penilaian skor menggunakan parameter variabel bebas data uji (X_test) dengan varaibel respon data uji (y_test). Implementasi pada kode Python sebagai berikut.

```
mlp.score(X_test, y_test)
```

Skor AUROC merepresentasi *confusion matrix* dari hasil prediksi data nyata 'y_test' terhadap data prediksi 'y_pred' dengan menghitung *False Positive Rate* (FPR) dan *True Positive Rate* (TPR). Skor AUROC juga dapat diukur terhadap nilai dari masing-masing variabel respon, seperti yang penulis terapkan pada uji coba, skor AUROC diukur terhadap probabilitas nilai 0 dan 1 dari variabel bebas data latih. Implementasi skor AUROC terhadap keseluruhan nilai variabel respon pada kode Python sebagai berikut.

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test, y_pred)
```

Untuk penilian skor AUROC terhadap masing-masing nilai variabel respon sebagai berikut.

```
y_prob = mlp.predict_proba(X_test)
y_prob_0 = [p[0] for p in y_prob]
y_prob_1 = [p[1] for p in y_prob]

roc_auc_score(y_test, y_prob_0)
roc_auc_score(y_test, y_prob_1)
```

Cohen kappa merupakan cara penilaian yang cukup penting dalam mengukur performa pemodelan klasifikasi, terutama pada kasus *imbalanced data*. Skor *cohen kappa* memberikan penjelasan bukan pada prediksi hasil distribusi tapi seberapa bagus performa klasifikasi yang dihasilkan dari pemodelan sehingga skor *cohen kappa* direpresentasikan dalam bentuk skala buruk (<0.4), cukup baik (0.4-0.75), dan sangat baik (0.75). Implementasi kode Python sebagai berikut.

```
from sklearn.metrics import cohen_kappa_score

cohen_kappa_score(y_test, y_pred)
```

3.2.6 Sensitivity dan Specificity

Distribusi hasil uji pemodelan menghasilkan matriks konfusi. Berdasarkan matriks konfusi dapat diketahui nilai dari TPR dan FPR. Untuk menentukan nilai TPR dan FPR berdasarkan nilai True Positive (TP), False Positive (FP), True Negative (TN) dan False Negative (FN).

Nilai TPR didapatkan dari TPR = TP / (TP + FN). Nilai TPR juga disebut sebagai *sensitivity*. Sedangkan nilai FPR didapatkan dari FPR = FP / (FP + TN). Nilai *specificity* didapatkan dari *specificity* = TN / (TN + FP).

Berdasarkan TPR dan FPR dapat juga dicari nilai *precision* dan *recall. Precision* didapatkan dari *precision* = TP / (TP + FP). Sedangkan *recall* bernilai sama dengan *sensitivity, recall* = TP / (TP + FN). Perhitungan *sensitivity, specificity, precission* dan *recall* digunakan untuk interprestasi pada kurva ROC / AUC. Kode pada program python :

```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_pred)
```

3.2.7 Implementasi pada aplikasi

Hasil dari analisis pemodelan ANN dikemas dalam aplikasi berbasis web menggunakan *Flask micro-framework* dan *database* Sqlite 3. Penerapan pada aplikasi bertujuan untuk memudahkan pengguna (dokter hewan) dalam melakukan perencanaan IB.

Aplikasi dirancang sesuai kebutuhan dokter hewan untuk menentukan wilayah peternak sapi yang diprediksi estrus. Data master yang digunakan adalah wilayah, peternak (anggota), sapi, transaksi IB, dan transaksi PKB. Dari hasil transaksi IB dan PKB, akan menambah atribut data sapi yang selanjutnya dilakukan penggabungan / *joining* untuk membuat tabel *view* dataset. Data pada tabel *view* dataset yang dijadikan sumber dalam perhitungan pemodelan. Kode pemodelan ditunjukkan sebagai berikut.

```
class Oversampling:
    df = getDataset()
    df.columns = range(df.shape[1])
    x =  df.loc[:,len(df.columns)-len(df.columns):len(df.columns)-
2]
    y = df.loc[:,len(df.columns)-1:len(df.columns)-1]
    major = y.loc[:,5].value_counts()[1]
    minor = y.loc[:,5].value_counts()[0]
    df_majority = df[df.loc[:,5]==1]
    df_minority = df[df.loc[:,5]==0]
```

```
    # Upsample minority class (Oversampling)
    df_minority_upsampled = resample(df_minority,
                                     replace=True,
                                     n_samples=major,
                                     random_state=1)
    # Combine majority class with upsampled minority class
    df = pd.concat([df_majority, df_minority_upsampled])

    x =  df.loc[:,len(df.columns)-len(df.columns):len(df.columns)-
2]
    y = df.loc[:,len(df.columns)-1:len(df.columns)-1]
    value = y.loc[:,5].value_counts()
    x_train, x_test, y_train, y_test = train_test_split(x,y,
                                                        test_size=.2,
                                                        random_state=1)
    def xtrain(self, x_train):
        return self.x_train
    def xtest(self, x_test):
        return self.x_test
    def ytrain(self, y_train):
        return self.y_train
    def ytest(self, y_test):
        return self.y_test
    def val(self, value):
        return self.value

class Classifier:
    xtr = MinMaxScaler().fit_transform(Oversampling.x_train)
    ytr = Oversampling.y_train
    mlp = MLPClassifier(hidden_layer_sizes=(6,8,6),
                        max_iter=1000,
                        activation='tanh',
                        random_state=1,
                        solver='lbfgs',
                        warm_start=False
                        )
    mlp.fit(xtr, ytr)

    xts = MinMaxScaler().fit_transform(Oversampling.x_test)
    yts = Oversampling.y_test
    y_pred = mlp.predict(xts)
    mlp_score = mlp.score(xts, yts)
    roc_score = roc_auc_score(yts, y_pred)
    cm = confusion_matrix(yts, y_pred)

    def cls(self, mlp):
        return self.mlp
    def mlpscore(self, mlp_score):
```

```
        return self.mlp_score
    def rocscore(self, roc_score):
        return self.roc_score
    def cfm(self, cm):
        return self.cm
```

Hasil dari pemodelan disimpan pada tabel prediksi yang memiliki kolom indikator nomor sapi, peternak dan wilayah yang digunakan acuan dokter hewan dalam melakukan perencanaan IB. Representasi dari aplikasi berupa list data sapi yang diprediksi mengalami estrus pada tiga hari semenjak perhitungan prediksi dilakukan. Apabila tabel 'prediksi' masih kosong, data hasil prediksi akan digenerate otomatis ketika mengakses halaman hasil prediksi. Jika sudah ada rekam data, akan dilakukan update untuk pengecekan update siklus estrus pada masing-masing sapi. Pseudocode ditunjukan pada kode berikut.

```
#kalkulasi prediksi
Input :
Output : hasil prediksi MLP Classifier
fungsi prediksi();
  for i = 0 in q <- jumlah record kueri tabel sapi :
    estrus <- prediksi_model_ANN([nilai min max setiap variabel])
    if (sapi_id pada tabel prediksi == NULL)
       record <- generate rekam data prediksi sapi_id
       simpan_sesi_database()
    elseif (sapi_id pada tabel prediksi == id sapi pada i)
       record <- update rekam data prediksi sapi_id
       sesi_database_update()
   end
simpan_kolektif_sesi_database()
```

Kustomisasi dalam kode python dengan mengatur parameter variabel ‘jarak_ib_sebelumnya’ dilakukan penjumlahan 3 hari seperti ditunjukkan pada kode berikut :

```
#kalkulasi prediksi
for i in q:
e = cls.mlp.predict([[
    (i.rpf - rpfmin)/(rpfmax-rpfmin),
    (i.perilaku - perilakumin)/(perilakumax-perilakumin),
    (i.ib_ke - ib_kemin)/(ib_kemax-ib_kemin),
    ((i.jarak_ib+3) - jarak_ibmin)/(jarak_ibmax-jarak_ibmin),
    (i.laktasi - laktasimin)/(laktasimax-laktasimin),]])

#kueri pada views
from sqlalchemy import text
q = text('SELECT
          sapi.no_sapi as "sapi",
          anggota.namaanggota as "anggota",
          prediksi.estrus "estrus",
          date(prediksi.updated_at, "+3 days") as "tanggal",
          wilayah.namawilayah as "wilayah"
          FROM prediksi LEFT JOIN sapi ON sapi.id =
          prediksi.sapi_id
          LEFT JOIN anggota ON anggota.id = sapi.anggota_id
          LEFT JOIN wilayah ON wilayah.id = anggota.wilayah_id
          WHERE prediksi.estrus = 1')
```