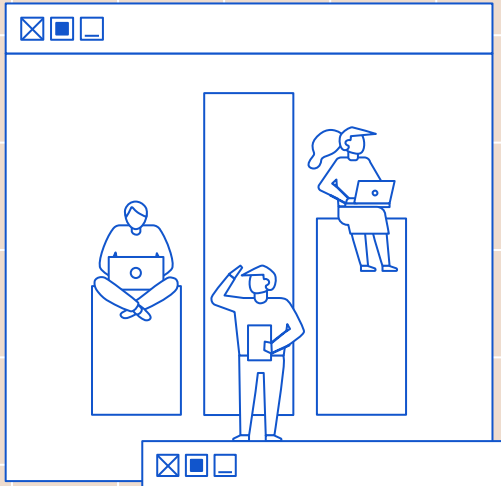
Analisis Akurasi dan Interpretasi **Model XGBoost** untuk **Pricing** Asuransi Kompensasi Pekerja

By Jonathan Stanley

Penelitian Evaluasi Model XGBoost



01

PENDAHULUAN

XGBoost u

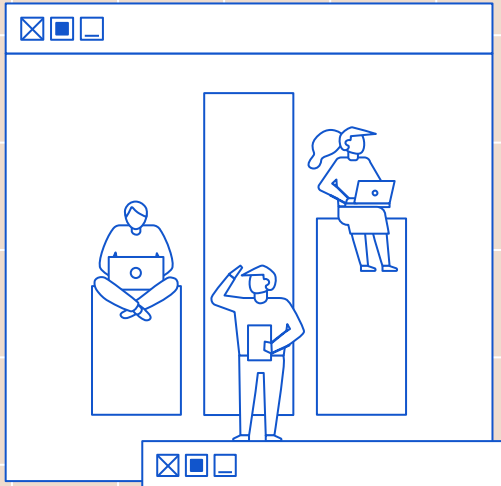
LATAR BELAKANG

Semua pekerjaan di sebagian besar negara wajib memiliki asuransi kompensasi pekerja. Asuransi kompensasi pekerja meliputi biaya pengobatan, kehilangan upah dan tunjangan lainnya jika seorang karyawan sakit saat bekerja.

Setiap pekerja tentu memiliki tingkat risiko yang berbeda-beda. Penilaian tingkat risiko klaim pemohon asuransi merupakan bagian penting dalam asuransi, sehingga perlu untuk diklasifikasikan. Penentuan tingkat risiko klaim pada asuransi didasarkan pada data historis pemohon. Pengajuan untuk menjadi anggota suatu asuransi membutuhkan waktu yang tidak singkat.

LATAR BELAKANG

Namun pengaplikasian suatu model machine learning dapat membantu mengklasifikasikan calon pemohon asuransi berdasarkan tingkat risiko dengan cepat. Salah satu model machine learning yaitu Extreme Gradient Boosting (XGBoost) yang merupakan suatu model berbasis decision tree. Model ini digunakan untuk memprediksi risiko pada asuransi. Adanya missing values pada data yang digunakan dapat diatasi dengan beberapa strategi pada proses pra pengolahan data untuk meningkatkan nilai akurasi model XGBoost. Jika dibandingkan dengan model decision tree, random forest dan Bayesian ridge, kinerja model XGboost masih unggul dalam memproses missing values pada data yang digunakan. Oleh sebab itu, pada final project ini, kami akan menganalisis akurasi dan juga menginterpretasi model XGBoost yang digunakan untuk penentuan premi (pricing) asuransi kompensasi pekerja.



02

LANDASAN TEORI

XGBoost u

DEFINISI XGBOOST

Metode XGBoost merupakan pengembangan dari metode Gradient Boosting. Gradient boosting merupakan algoritma yang dapat menemukan solusi yang optimal untuk masalah regresi, klasifikasi dan ranking. Konsep dasar dari algoritma ini adalah menyesuaikan parameter pembelajaran secara berulang untuk menurunkan loss function (mekanisme evaluasi atas model). XGBoost menggunakan model yang lebih teratur untuk membangun struktur pohon regresi, sehingga memberikan kinerja yang lebih baik, dan mengurangi kompleksitas model untuk menghindari overfitting. Hasil prediksi akhir dari XGBoost adalah penjumlahan hasil prediksi dari setiap pohon regresi. Algoritma berbasis decision tree memiliki kinerja yang baik pada data dengan fitur kategorikal dan tidak terlalu berpengaruh terhadap data dengan kelas tidak seimbang.

FUNGSI OBJEKTIF

Pada metode ini diperlukan fungsi objektif yang berguna untuk menilai seberapa bagus model yang didapatkan sesuai dengan data latih. Karakteristik yang terpenting dari fungsi objektif terdiri dari 2 bagian yaitu nilai pelatihan yang hilang dan nilai regularisasi dengan persamaan

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

Dimana L adalah fungsi pelatihan yang hilang, dan Ω adalah fungsi regularisasi, dan θ adalah parameter model terkait. Fungsi pelatihan yang hilang secara umum dapat ditulis seperti pada persamaan

$$L(\theta) = \sum_{i=1}^{n} l\left(y_i, \widehat{y_i}\right)$$

Dimana yi adalah nilai data sebenarnya yang dianggap benar dan $yi\ \hat{}$ adalah hasil nilai prediksi dari model, sedangkan n adalah jumlah iterasi nilai dari model.

RELATED WORKS

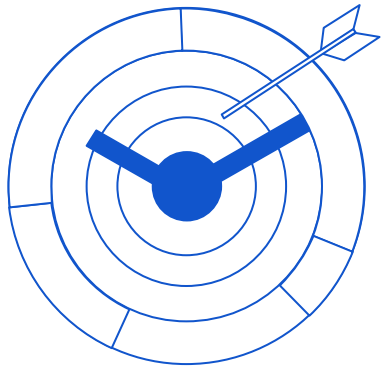
1) Skripsi Penelitian Analisis Algoritma Image Generator for Tabular Data untuk Prediksi Klaim Asuransi Kompensasi Pekerja Menggunakan Convolutional Neural Network oleh Reri Nandar Munazat

Penelitian tersebut menguji akurasi model CNN berbasis algoritma IGTD dalam memprediksi klaim asuransi kompensasi pekerja serta membandingkan performa model tersebut dengan model Multi-Layer Perceptron, Random Forest, serta Xtreme Gradient Boosting. Hasil simulasi dengan metode repeated holdout sebanyak lima iterasi menunjukkan bahwa model CNN dapat memprediksi klaim dengan baik meskipun secara umum belum mampu menyaingi model-model non-CNN secara signifikan.

TUJUAN PENELITIAN

1) Menentukan model dengan hasil terbaik untuk pricing asuransi kompensasi pekerja menggunakan model XGboost

2) Mendapatkan hasil interpretasi dari model XGBoost terhadap masalah pricing asuransi kompensasi pekerja



TAHAPAN METODE PENELITIAN

Pemrosesan Data

Sumber data publik mengenai Asuransi Kompensasi Pekerja

Modelling

Proses Modelling XGBoost

Evaluasi

Evaluasi performa model menggunakan metrik evaluasi

Data Processing

Data Collecting : Menggunakan dataset publik yang tersedia di Kaggle, yaitu dataset Actuarial loss prediction mengenai asuransi kompensasi pekerja.

Data Understanding : Memahami setiap variabel pada dataset serta mendapatkan pemahaman yang lebih lanjut untuk menarik insight dengan menggali data lebih dalam menggunakan visualisasi, dan EDA.

Data Preprocessing : Melakukan filter sehingga data yang tidak relevan dihilangkan, serta melakukan pembersihan data untuk missing values, invalid data, unstructured data, imbalance data, dan masalah data lainnya.

Data Transformation : Merubah data menjadi bentuk yang terstruktur sebelum modeling, dimana pada regresi XGBoost data akan diubah menjadi bentuk numerik menggunakan *one hot encoding*.

Data Modelling

Data Splitting : Membagi data menjadi dua bagian, yaitu data training dan data testing.

Model Building : Membuat model XGBoost menggunakan pilihan model regresi menggunakan *XGBRegressor*

Optimisation : Mencari hyperparameter yang terbaik untuk model menggunakan Bayes Search dan juga K-fold Cross Validation

Model Prediction: Membuat prediksi menggunakan Model XGBoost dengan menggunakan testing data

Data Evaluation

A) Evaluation Metrics

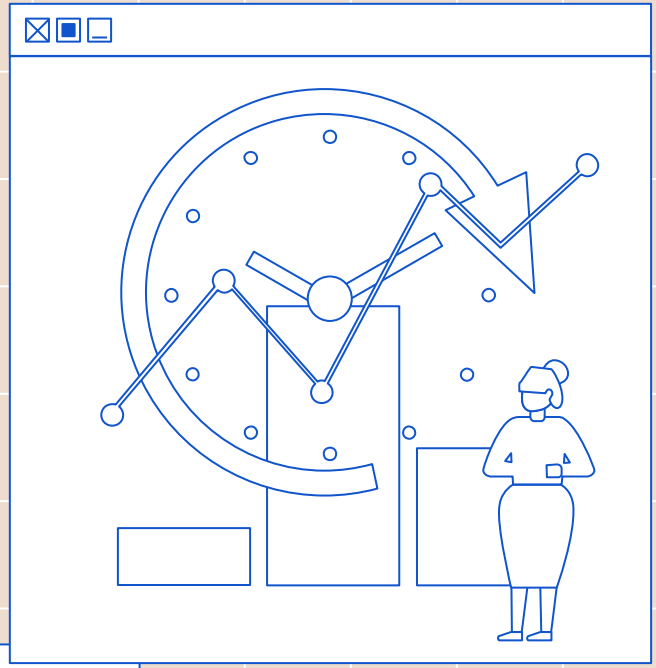
- **Mean Absolute Error (MAE)**
- **Root Mean Squared Error(RMSE)**
- **R-Squared (R^2)**
- **Coverage Probability**

B) Model Interpretation

Melakukan interpretasi terhadap model XGBoost untuk pricing asuransi kompensasi pekerja dengan cara mengevaluasi seberapa pentingnya variabel dalam mempengaruhi hasil prediksi (**Feature Importance**)

03

SIMULASI DAN ANALISIS HASIL



Simulasi Pembuatan Model XGBoost

DATASET

Dataset yang digunakan adalah ‘**Actuarial loss prediction**’ , sebuah dataset publik yang diperoleh melalui Kaggle

Dataset ini mencakup 90.000 kebijakan asuransi kompensasi pekerja yang realistis, yang semuanya mengalami kecelakaan. Untuk setiap catatan, terdapat informasi demografis dan terkait pekerja, serta deskripsi teks mengenai kecelakaan tersebut.

DATASET INFORMATION

Data fields (15 fields)

- ClaimNumber: Identifikasi unik kebijakan
- DateTimeOfAccident: Tanggal dan waktu kecelakaan
- DateReported: Tanggal pelaporan kecelakaan
- Age: Usia pekerja
- Gender: Jenis kelamin pekerja
- MaritalStatus: Status pernikahan pekerja. (M)enikah, (S)ingle, (U)nknown.
- DependentChildren: Jumlah anak tanggungan
- DependentsOther: Jumlah tanggungan lainnya selain anak-anak
- WeeklyWages: Total upah mingguan
- PartTimeFullTime: Biner (P) atau (F)
- HoursWorkedPerWeek: Total jam kerja per minggu
- DaysWorkedPerWeek: Jumlah hari kerja per minggu
- ClaimDescription: Deskripsi bebas teks tentang klaim
- InitialIncurredClaimCost: Estimasi awal oleh perusahaan asuransi mengenai biaya klaim
- UltimateIncurredClaimCost: Total pembayaran klaim oleh perusahaan asuransi.

DATA READING

train.csv

- Dataset untuk training model dengan lebih dari 54.000 insurance policies

Mengubah Fitur DateTime

```
#pengubahan tipe fitur DateTime & seleksi data
df['DateTimeOfAccident'] = pd.to_datetime(df['DateTimeOfAccident'])
df['DateReported'] = pd.to_datetime(df['DateReported'])
df['YearReported'] = df['DateReported'].dt.year
```

DATA UNDERSTANDING

- **5 Baris Pertama dari dataframe**

ClaimNumber	WC8285054	WC6982224	WC5481426	WC9775968	WC6828422
DateTimeOfAccident	2002-04-09 07:00:00+00:00	1999-01-07 11:00:00+00:00	1996-03-25 00:00:00+00:00	2005-06-22 13:00:00+00:00	1999-06-21 11:00:00+00:00
DateReported	2002-07-05 00:00:00+00:00	1999-01-20 00:00:00+00:00	1996-04-14 00:00:00+00:00	2005-07-22 00:00:00+00:00	1999-09-09 00:00:00+00:00
Age	48	43	30	41	50
Gender	M	F	M	M	M
MaritalStatus	M	M	U	S	M
DependentChildren	0	0	0	0	0
DependentsOther	0	0	0	0	0
WeeklyWages	500.0	509.34	709.1	555.46	200.0
PartTimeFullTime	F	F	F	F	F
HoursWorkedPerWeek	38.0	37.5	38.0	38.0	38.0
DaysWorkedPerWeek	5	5	5	5	5
ClaimDescription	LIFTING TYRE INJURY TO RIGHT ARM AND WRIST INJURY	STEPPED AROUND CRATES AND TRUCK TRAY FRACTURE ...	CUT ON SHARP EDGE CUT LEFT THUMB	DIGGING LOWER BACK LOWER BACK STRAIN	STRUCK HEAD ON HEAD LACERATED HEAD
InitialIncurredCalimsCost	1500	5500	1700	15000	500
UltimateIncurredClaimCost	4748.203388	6326.285819	2293.949087	17786.48717	598.762315
YearReported	2002	1999	1996	2005	1999

DATA UNDERSTANDING

- **Statistical Descriptive dari data numerik pada dataframe**

	count	mean	std	min	25%	50%	75%	max
Age	45291.0	34.009207	12.159998	13.000000	24.00000	32.000000	43.000000	81.000
DependentChildren	45291.0	0.079861	0.418868	0.000000	0.00000	0.000000	0.000000	9.000
DependentsOther	45291.0	0.007397	0.090693	0.000000	0.00000	0.000000	0.000000	3.000
WeeklyWages	45291.0	428.014364	260.023794	1.000000	200.00000	412.770000	521.180000	7497.000
HoursWorkedPerWeek	45291.0	37.569711	12.452778	0.000000	38.00000	38.000000	40.000000	640.000
DaysWorkedPerWeek	45291.0	4.894063	0.581220	1.000000	5.00000	5.000000	5.000000	7.000
InitialIncurredCalimsCost	45291.0	8753.829922	22102.047855	1.000000	756.00000	2500.000000	10000.000000	2000000.000
UltimateIncurredClaimCost	45291.0	11947.669219	35420.756715	121.886805	945.02231	3723.166477	9116.191048	4027135.935
YearReported	45291.0	1998.066106	4.352081	1991.000000	1994.00000	1998.000000	2002.000000	2006.000

DATA PREPROCESSING

Missing Values

- Mencari kolom yang memiliki missing values

```
# Hitung missing values
df.isnull().sum()
```

✓ 0.0s

```
DateTimeOfAccident            0
DateReported                  0
Age                           0
Gender                        0
MaritalStatus                28
DependentChildren             0
DependentsOther               0
WeeklyWages                   0
PartTimeFullTime              0
HoursWorkedPerWeek            0
DaysWorkedPerWeek             0
ClaimDescription              0
InitialIncurredCalimsCost     0
UltimateIncurredClaimCost     0
YearReported                  0
dtype: int64
```

Terdapat missing value pada field ‘MaritalStatus’ yaitu sebanyak 28 values pada baris yang berbeda.

DATA PREPROCESSING

Feature Selection

```
#penghapusan data dengan nilai fitur tidak realistis
df = df.loc[(df['HoursWorkedPerWeek'] <= 168)&(df['HoursWorkedPerWeek']>0)&
           (df['ReportingDelay']>=0)]
```

```
#penghapusan unknown value

df.dropna(inplace=True)
df = df.loc[(df['Gender'] != 'U')&(df['MaritalStatus'] != 'U')]
```

DATA PREPROCESSING

Missing Values

- ‘MartialStatus’ akan di analisa lebih lanjut dan missing values akan ditangani

MaritalStatus	mean	median	count
M	13028.654077	4522.489306	18654
S	9857.401085	2486.967424	21926
U	17466.998886	6469.310541	4683

'MartialStatus' memiliki value : (M)arried, (S)ingle, (U)nknown. Missing values pada kolom ini akan disubtitusikan dengan nilai **U .**

```
# Mengisi missing values dengan status U = Unknown
df['MaritalStatus']= df['MaritalStatus'].fillna('U')
```

✓ 0.0s

DATA PREPROCESSING

Feature Engineering

- Membuat field baru ‘ReportingDelay’ yaitu jarak waktu dari waktu kejadian di laporkan dan waktu dari kejadian.
- Menghapus Field ‘ClaimDescription’ karena valuenya tidak relevan untuk prediksi.

```
#pengubahan tipe fitur DateTime

df['DateTimeOfAccident'] = pd.to_numeric(df['DateTimeOfAccident'])
df['DateReported'] = pd.to_numeric(df['DateReported'])

#pemilihan dan rekayasa fitur
df['ReportingDelay'] = (df['DateReported'] - df['DateTimeOfAccident'])
df.drop(columns=['ClaimDescription'], inplace=True)
```

DATA TRANSFORMATION

- Mengganti field dengan value kategorik dengan nilai dummy menggunakan One Hot Encoding
- Mendefinisikan target variabel
- Melakukan Normalisasi pada data

```
#One Hot Encoding
category = pd.get_dummies(df[['Gender','MaritalStatus','PartTimeFullTime']],
                          drop_first=True)
#Target variabel
label = df[['UltimateIncurredClaimCost']]
df.drop(columns=['DateReported','Gender','MaritalStatus','PartTimeFullTime',
                 'UltimateIncurredClaimCost','YearReported'], inplace=True)
df = pd.concat([df, category], axis=1)

#normalisasi
scaler = MinMaxScaler()
df[df.columns] = scaler.fit_transform(df)
df
```

DATA MODELLING

XGBRegressor

Fungsi gradient boosting dengan algoritma yang digunakan untuk regresi

Bayesian Optimization

Teknik optimasi untuk mencari hyperparameters yang terbaik untuk performa model

Hyperparameters

- alpha : Untuk regularisasi L1 **(0.01 – 10)**
- learning_rate : Ukuran langkah setiap iterasi boosting **(0.01 – 0.5)**
- max_depth : Maximum depth dari setiap decision tree **(2 – 10)**
- n_estimators : Jumlah iterasi boosting **(50 – 1000)**
- min_child_weight : Jumlah minimum dari beban yang dibutuhkan pada node **(1 – 10)**
- subsample : Proporsi dari data pelatihan **(0.5 – 1.0)**
- colsample_bytree : Proporsi dari fitur yang dipilih secara acak **(0.1 – 1.0)**
- gamma : Minimum loss reduction yang dibutuhkan **(0 – 10)**

```
# Mendefinisikan Jangkauan searching untuk hyperparameter
params = {
    "alpha": Real(0.01, 10, prior='log-uniform'),
    "learning_rate": Real(0.01, 0.5, prior='uniform'),
    "max_depth": Integer(2, 10),
    "n_estimators": Integer(50, 1000),
    "min_child_weight": Real(1, 10),
    "subsample": Real(0.5, 1.0, prior='uniform'),
    "colsample_bytree": Real(0.1, 1.0, prior='uniform'),
    "gamma": Real(0, 10)
}
```

Evaluation Metrics

- **Root Mean Squared Error**

Mengukur akar dari rata-rata kuadrat kesalahan antara nilai prediksi dan nilai aktual pada skala yang sama. Semakin rendah nilai RMSE, semakin baik performa model

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

- **Mean Absolute Error**

Mengukur rata-rata kesalahan absolut antara nilai prediksi dan nilai aktual pada skala yang sama. Semakin rendah nilai MAE, semakin baik performa model

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Evaluation Metrics

- **R - Squared (R^2)**

  Mengukur seberapa dekat nilai prediksi dengan nilai aktual dalam skala 0 – 1.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

  * Dimana :

  n = banyaknya data (observasi)

  yi = nilai observasi ke-i

  $\hat{y}i$ = nilai prediksi observasi ke-i

  $\bar{y}$ = rata - rata nilai observasi

Evaluation Metrics

- **Coverage Probability**

Mengukur sejauh mana residual(selisih nilai aktual dan prediksi) berada dalam interval kepercayaan yang ditetapkan. Metrik ini merepresentasikan probabilitas nilai residual yang distandarisasi berada dalam jangkauan kuantil atas dan bawah dari distribusi normal

Perhitungan Matematika :**coverage_prob = avg(P(norm.ppf(alpha / 2) < std_residuals < norm.ppf(1 - alpha / 2)))**

Perhitungan ini akan menguji apakah nilai residual berada dalam jangkauan probabilitas kuantil bawah dan atasnya, hasil dari pengujian tersebut berupa boolean, dimana ‘1’ menunjukkan True dan ‘0’ menunjukkan False. Hasil dari seluruh pengujian akan dirata-ratakan untuk mendapatkan nilai coverage probability

* Dimana :

residuals = $|\hat{y}i - yi|$

P = probabilitas

norm.ppf = percent point function

std_residuals = nilai residual yang telah di standarisasi

alpha = tingkat signifikansi

DATA MODELLING

Modelling Process

Modelling akan dilakukan sebanyak 10 iterasi menggunakan data yang di split secara berbeda dengan random_state 0 sampai 9 dan akan diambil nilai rata-rata dari setiap metrik evaluasi dari 10 iterasi.

```
max_iteration = 10

for iteration in range(max_iteration):
    print(f"Iteration {iteration+1}")

    # Splitting data training dan testing dengan perbandingan 70:30.
    X_train_full, X_test, y_train_full, y_test = train_test_split(X, y, test_size=0.3, random_state=iteration)
```

DATA MODELLING

4-fold Cross Validation

Pada setiap iterasi, data training akan dibagi menjadi 4 folds, dimana model akan dilatih pada 3 folds yang telah dibuat, dan dievaluasi pada 1 fold yang tersisa untuk performanya.

```
# k-fold Cross Validation
k = 4
kf = KFold(n_splits=k, shuffle=True, random_state=0)
```

```
for train_index, val_index in kf.split(X_train_full):
    X_train, X_val = np.array(X_train_full)[train_index], np.array(X_train_full)[val_index]
    y_train, y_val = np.array(y_train_full)[train_index], np.array(y_train_full)[val_index]
```

Fungsi Evaluasi

```python
for train_index, val_index in kf.split(X_train_full):
    X_train, X_val = np.array(X_train_full)[train_index], np.array(X_train_full)[val_index]
    y_train, y_val = np.array(y_train_full)[train_index], np.array(y_train_full)[val_index]

    # Objective Function yang terdefnisi dalam loop
    def objective_function(alpha, learning_rate, max_depth, n_estimators, min_child_weight, subsample, colsample_bytree, gamma):
        model = xgb.XGBRegressor(
            alpha=alpha,
            learning_rate=learning_rate,
            max_depth=int(max_depth),
            n_estimators=int(n_estimators),
            min_child_weight=min_child_weight,
            subsample=subsample,
            colsample_bytree=colsample_bytree,
            gamma=gamma
        )

        model.fit(X_train, y_train)
        y_pred = model.predict(X_val)

        return -mean_squared_error(y_val, y_pred)
```

Fungsi dimana hyperparameters akan digunakan untuk pelatihan model XGboost Regressor dan melakukan fitting pada training data serta evaluation data untuk evaluasi

Bayes Search

```
# Bayes Search
bayes_opt = BayesSearchCV(
    xgb.XGBRegressor(),
    params,
    n_iter=50,
    cv=kf,
    scoring='neg_mean_squared_error',
    random_state=0,
    verbose=0,
    fit_params={
        'eval_set': [(X_val, y_val)],
        'early_stopping_rounds': 10,
        'verbose': 1
    }
)

bayes_opt.fit(X_train, y_train)
best_params = bayes_opt.best_params_
```

Akan dilakukan proses bayes search dengan fungsi dan parameter yang telah ditentukan sebelumnya menggunakan set evaluasi. Proses optimasi akan dilakukan **50** iterasi untuk mendapatkan hyperparameter terbaiknya. Terdapat 'early_stopping_rounds' : 10 yang akan menghentikan proses pencarian jika tidak ada peningkatan hasil setelah **10** iterasi.

Training and Fitting

Normal Distribution

Tweedie Distribution

Proses Modelling akan dibagi menjadi 2 kondisi, yaitu kondisi dimana model dibuat dengan asumsi data berdistribusi normal, dan juga data berdistribusi tweedy. Hasil dari keduanya akan dibandingkan dan dipilih yang terbaik.

Normal Distribution

```python
# Pelatihan Model XGBoost dengan parameter terbaik
model = xgb.XGBRegressor(
    alpha=best_params['alpha'],
    learning_rate=best_params['learning_rate'],
    max_depth=best_params['max_depth'],
    n_estimators=best_params['n_estimators'],
    min_child_weight=best_params['min_child_weight'],
    subsample=best_params['subsample'],
    colsample_bytree=best_params['colsample_bytree'],
    gamma=best_params['gamma']
)
# Fitting model dengan data training dan evaluation set
model.fit(X_train, y_train,
          eval_set=[(X_val, y_val)],
          early_stopping_rounds=10,
          verbose=1)

y_pred = model.predict(X_test)
```

Model akan dilatih menggunakan hyperparameter terbaik yang sebelumnya didapatkan menggunakan bayesian search dan akan dilakukan fitting menggunakan data pelatihan serta prediksi menggunakan data testing dengan asumsi data berdistribusi normal

Tweedie Distribution

```python
# Pelatihan Model XGBoost dengan parameter terbaik
model = xgb.XGBRegressor(
    alpha=best_params['alpha'],
    learning_rate=best_params['learning_rate'],
    max_depth=best_params['max_depth'],
    n_estimators=best_params['n_estimators'],
    min_child_weight=best_params['min_child_weight'],
    subsample=best_params['subsample'],
    colsample_bytree=best_params['colsample_bytree'],
    gamma=best_params['gamma'],
    objective="reg:tweedie",
    tweedie_variance_power=1.5
)
# Fitting model dengan data training
model.fit(X_train, y_train,
          verbose=1)

y_pred = model.predict(X_test)
```

Kondisi dimana data berdisitrubsi tweedie dan digunakan tweedie_variance_power = 1.5 yang umum digunakan dalam kasus insurance

Model Evaluation

```
# Metrik Evaluasi Umum
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
# Variabel untuk menghitung Coverage probability dan VaR
alpha = 0.05
residuals = y_test - y_pred
std_residuals = residuals / np.std(residuals)

# Metrik Coverage Probability
coverage_prob = np.mean((std_residuals > norm.ppf(alpha / 2)) & (std_residuals < norm.ppf(1 - alpha / 2)))
```

Metrik Evaluasi **RMSE**, **MAE**, **R^2, dan coverage probabilities** dari setiap iterasi akan dihitung untuk setiap iterasi dari proses running modelnya.

Model Evaluation

```
    # Append metrik evaluasi
    rmse_iteration_scores.append(rmse)
    mae_iteration_scores.append(mae)
    r2_iteration_scores.append(r2)
    coverage_prob_iteration_scores.append(coverage_prob)


# Average metrik untuk setiap iterasi
best_rmse = np.mean(rmse_iteration_scores)
best_mae = np.mean(mae_iteration_scores)
best_r2 = np.mean(r2_iteration_scores)
best_coverage_prob = np.mean(coverage_prob_iteration_scores)
```

Metrik Evaluasi akan disimpan dalam sebuah list untuk setiap loop yang telah diselesaikan dan akan dicari rata-rata setiap metrik evaluasi dari 10 pengulangan

Simulasi Model Normal Distribution (XGBoost)

Iterasi	RMSE	MAE	R^2	Coverage Probabilities
1	25968.69238	7776.551897	0.298341181	0.962100571
2	21866.07138	7312.910485	0.351262468	0.956356364
3	24345.55286	7679.684694	0.301771925	0.957621672
4	24414.66485	7642.049827	0.320861983	0.959365816
5	23001.95158	7517.913753	0.330120825	0.957953967
6	25948.37394	7572.614201	0.297586601	0.963609381
7	22563.73819	7492.145842	0.341246022	0.956597847
8	23472.38883	7578.112315	0.332295237	0.95852725
9	22524.28331	7148.837104	0.315541598	0.957836856
10	24569.32344	8043.503035	0.296684382	0.960870239

Simulasi Model Tweedie Distribution (XGBoost)

Iterasi	RMSE	MAE	R^2	Coverage Probabilities
1	29868.93235	8613.426131	0.065555526	0.96413423
2	25300.39204	7422.703549	0.125983121	0.962907371
3	29014.63958	8564.035081	0.007058763	0.96301707
4	29657.90327	9304.605827	-0.007985353	0.964153027
5	24444.82612	6977.792033	0.242775641	0.965933196
6	30265.64652	8068.274172	0.04282584	0.966938086
7	25151.75384	7644.174611	0.172565175	0.964096738
8	28297.53554	9217.830607	0.019824674	0.958492956
9	25976.17659	8019.902489	0.082169389	0.961135356
10	28028.09387	8387.272671	0.078810928	0.96252687

Evaluation Result

- Normal Distribution (XGBoost)

Evaluation Metrics	RMSE	MAE	R^2	Coverage Probability
Mean	23,867.50	7,576.43	0.31857	0.95908
Standard Deviation	1,348.35	231.52	0.01878	0.00227

Evaluation Result

- Tweedie Distribution (XGBoost)

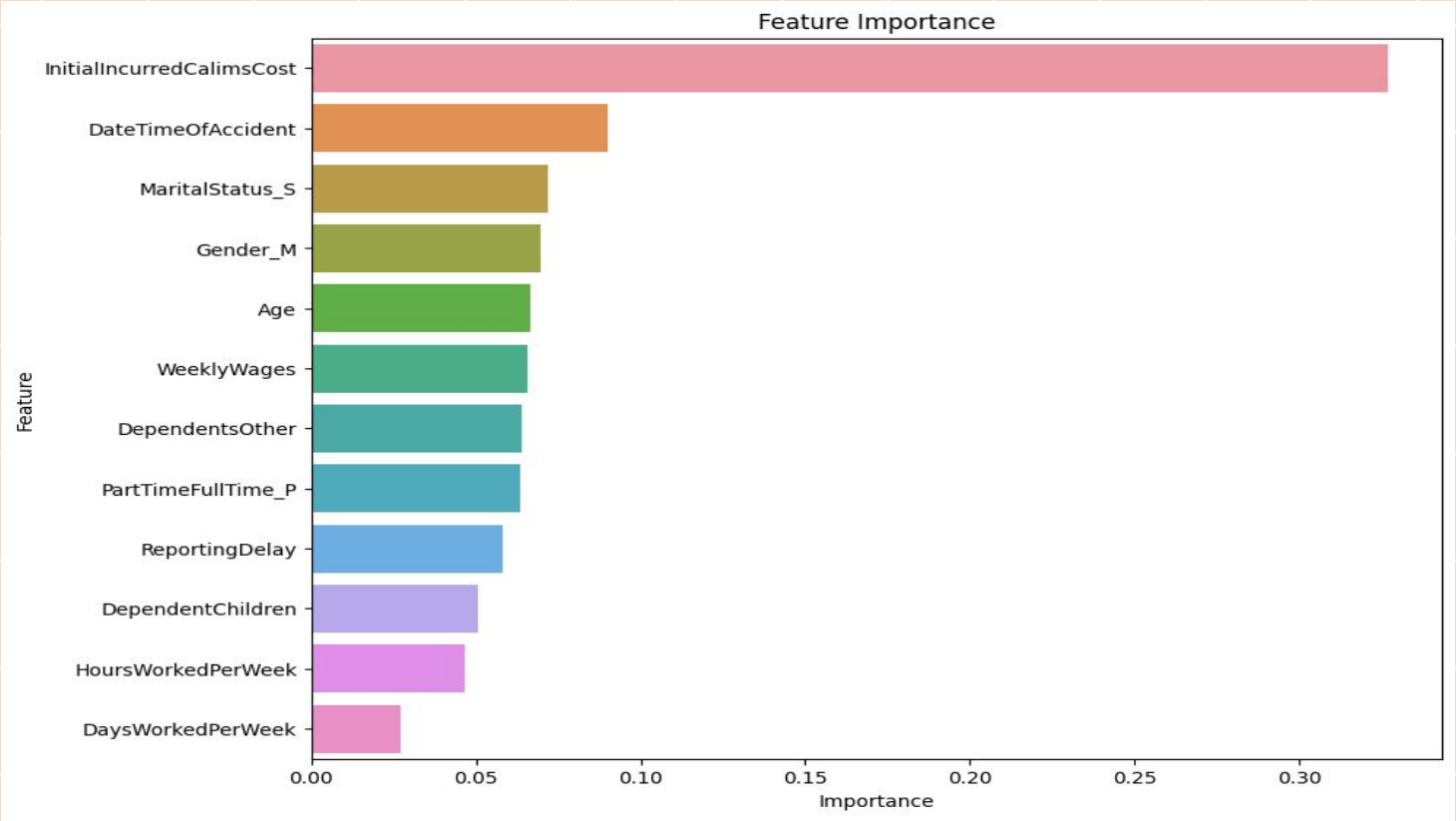
Evaluation Metrics	RMSE	MAE	R^2	Coverage Probability
Mean	27,600.59	8,222.01	0.08296	0.96333
Standard Deviation	2,074.46	710.39	0.07441	0.00225

Result and Discussion

Model XGBoost dengan distribusi Normal memperoleh hasil yang lebih baik dibandingkan model dengan distribusi Tweedie. Model yang dipilih untuk perbandingan dengan metode lain adalah model XGBoost dengan distribusi Normal

Coverage Probability dengan nilai mean yang tinggi mendekati satu (0.95908) dan standar deviasi yang rendah (0.00227) menunjukkan bahwa model sudah bekerja dengan baik untuk menangkap ketidakpastian dalam prediksi dan memberikan estimasi yang dapat diandalkan

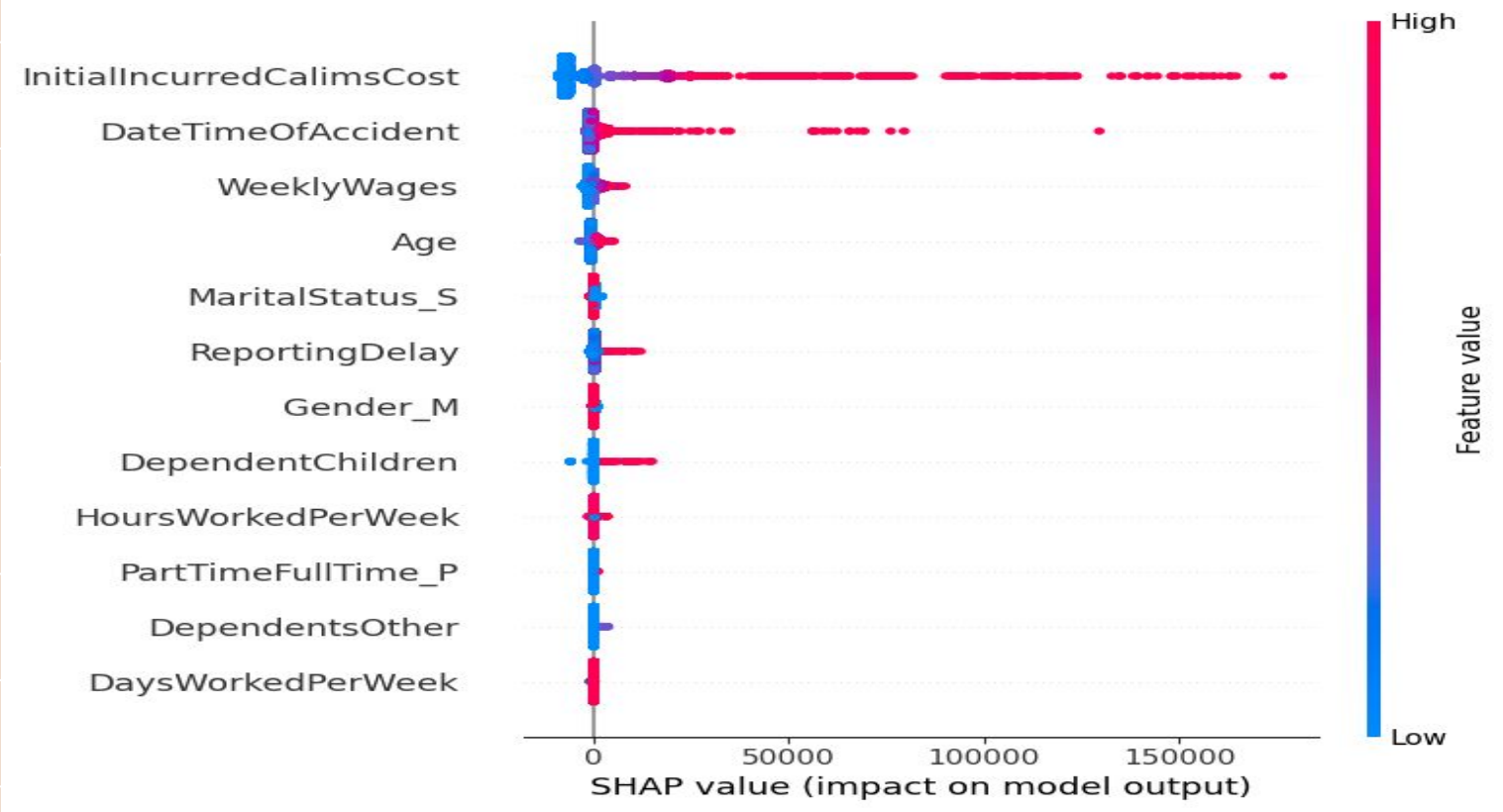
Feature Importance



Feature Importance

- Feature Importance dinilai berdasarkan kemampuan setiap variabel untuk mengurangi error pada saat model melakukan prediksi secara keseluruhan
- Variabel yang paling besar pengaruhnya terhadap hasil prediksi model adalah ‘InitialIncurredCalimsCost’ yang melebihi tingkat importance 0.3 dari total keseluruhannya, diikuti oleh ‘DateTimeOfAccident’ dengan tingkat importance sebesar kurang lebih 0.1
- Variabel yang paling kecil pengaruhnya terhadap hasil prediksi model adalah ‘DaysWorkedPerWeek’ yang kurang lebih memiliki tingkat importance 0.025 dari total keseluruhannya.
- Variabel lain diluar ketiga variabel yang telah disebutkan diatas memiliki tingkat importance yang relatif dekat nilainya dengan yang lain yaitu dalam jangkauan 0.05 - 0.08

SHapley Additive Explanation(SHAP)

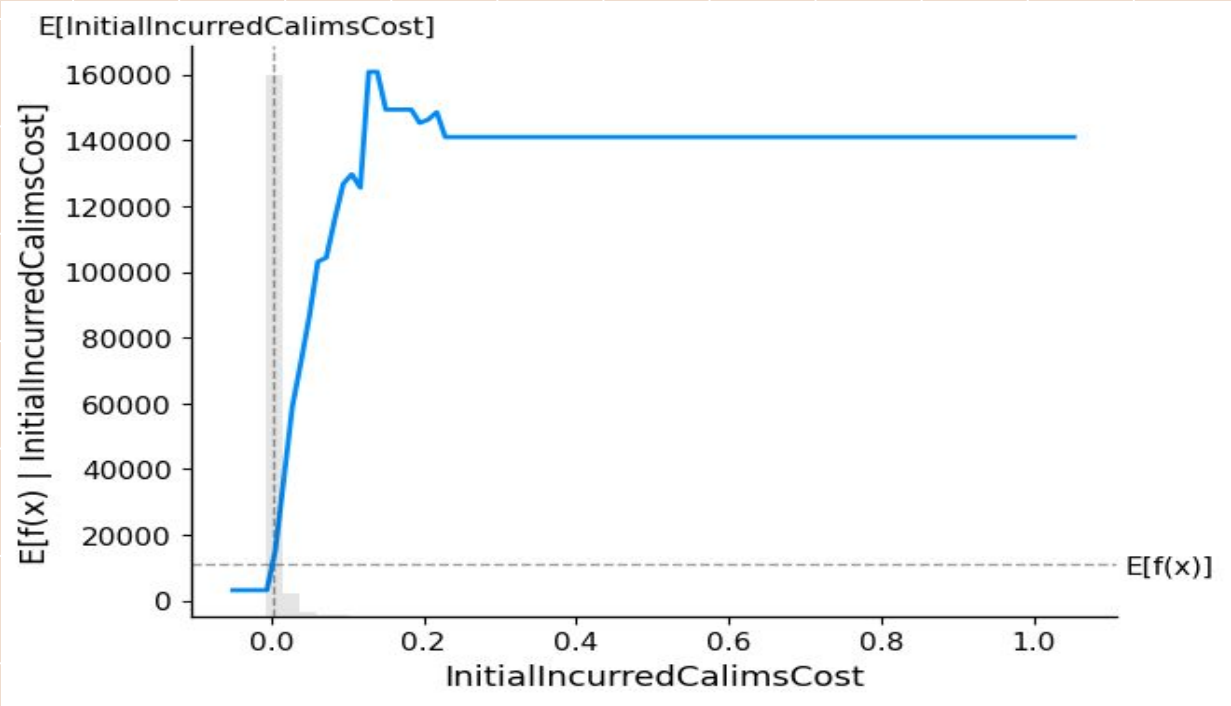


SHapley Additive Explanation(SHAP)

- SHAP menilai kontribusi setiap variabel dalam memprediksi dengan fokus untuk setiap variabel secara individu dengan cara lebih memfokuskan bagaimana masing-masing dampak dari variabel dibandingkan secara keseluruhannya
- Dari plot SHAP ini, terlihat bahwa ‘InitialIncurredCalimsCost’ memiliki pengaruh individu yang paling besar terhadap prediksi model, namun berbeda dengan Feature Importance, variabel-variabel lainnya memiliki urutan yang berbeda dalam seberapa besar pengaruh masing-masing variabel jika dinilai secara individu

Partial Dependance Plot

- Dibuat Partial Dependance Plot untuk variabel yang memiliki pengaruh paling besar (‘InitialIncurredCalimsCost’)



Partial Dependance Plot

- Partial Dependance Plot (PDP) menggunakan nilai dari kolom ‘InitialIncurredClaimCost’ yang telah dinormalisasi.
- Menurut PDP yang telah dibuat, total pembayaran klaim asuransi mengalami peningkatan drastis pada permulaan peningkatan nilai dari ‘InitialIncurredClaimCost’ yang nilai tertingginya berada pada kurang lebih 0.17 (Setelah dinormalisasi) dan kemudian mulai menurun secara perlahan.
- Setelah mencapai threshold tertentu (kurang lebih 0.28) untuk ‘InitialIncurredClaimCost’ maka tidak ada perubahan pada target total pembayaran klaim asuransi (dimana artinya jika sudah mencapai batas tertentu tidak ada perubahan yang signifikan untuk hasilnya, hal ini bisa terjadi karena kekurangan data informasi pada rentang tersebut).

Summary

- Model XGBoost sudah cukup baik dalam memprediksi pembayaran klaim oleh perusahaan asuransi berdasarkan metrik evaluasi RMSE, MAE, dan juga R-squared.

- Dalam dataset Actuarial Loss Prediction, distribusi normal lebih cocok digunakan dibandingkan distribusi tweedie untuk pembuatan Model XGBoost.

- Walaupun hasil evaluasi model XGBoost ini sudah cukup baik, performa model masih bisa ditingkatkan lebih lanjut dengan berbagai metode lain seperti Ensemble Learning untuk menggabungkan beberapa model XGBoost dalam proses modelling.

References

- Dataset : https://www.kaggle.com/c/actuarial-loss-estimation
-

THANK YOU

Jonathan Stanley

jonathanstanleyofficial@gmail.com