🔬 Sesi 4

Teknik Analisis Data Lanjutan

HARI 2: Analisis Lanjutan & Dashboard

Durasi: 3 jam (08:00 - 12:00)
Notebooks:

- `01_data_cleaning.ipynb`
- `02_time_series.ipynb`
- `03_statistical_analysis.ipynb`

🎯 Tujuan Sesi

Setelah sesi ini, Anda bisa:

- ✅ Menangani missing values dengan berbagai strategi
- ✅ Mendeteksi dan treatment outliers
- ✅ Feature engineering untuk analisis lanjutan
- ✅ Time series analysis dan trend forecasting
- ✅ Statistical hypothesis testing
- ✅ A/B testing dasar-dasar
- ✅ Data perubahan teknik

📋 Agenda Sesi

Waktu	Topik	Durasi
08:00 - 08:15	Kick-off & Problem Framing	15 min
08:15 - 09:00	Data Cleaning & Transformation	45 min
09:00 - 09:45	Feature Engineering & Outliers	45 min
09:45 - 10:30	Time Series Analysis	45 min
10:30 - 11:15	Forecasting Mini Lab	45 min
11:15 - 12:00	Statistical Analysis & Wrap-up	45 min

🧹 Part 1: Data Cleaning & Data Transformation

Kenapa penting? Data mentah biasanya kotor: ada yang kosong, salah, atau duplikat
Kita perlu bersihkan dulu sebelum analisis

Notebook: `01_data_cleaning.ipynb`

Kenapa pakai Data Cleaning?

- Real-world data is messy 🗑️
- Missing values, duplicates, errors
- 80% of data science is data preparation
- "Garbage in, garbage out"

🧹 Part 1: Data Cleaning & Data Transformation (lanjutan)

Yang akan dipelajari:

- Missing values detection & treatment
- Outlier detection & handling
- Data transformation
- Feature engineering

🕵️ Missing Values: Detection

```python
# Check missing values
missing = df.isnull().sum()
print(missing[missing > 0])

# Percentage missing
missing_pct = (df.isnull().sum() / len(df) * 100)
print(missing_pct[missing_pct > 0])

# Visualize missing data
import missingno as msno  # pip install missingno
msno.matrix(df)
plt.show()

# Heatmap korelasi missing values
msno.heatmap(df)
plt.show()
```

🛠️ Missing Values: Treatment Strategies

1. Drop Missing

```python
# Drop rows dengan missing
df_clean = df.dropna()

# Drop specific columns
df_clean = df.dropna(
    subset=['pagu', 'metode']
)

# Drop jika terlalu banyak missing
df_clean = df.dropna(
    thresh=len(df.columns)*0.7
)
```

Kapan pakai: < 5% missing, random pattern

2. Fill Missing

```python
# Fill dengan konstanta
df['col'].fillna(0)

# Fill dengan statistik
df['pagu'].fillna(
    df['pagu'].mean()
)

# Forward/Backward fill
df.fillna(method='ffill')
df.fillna(method='bfill')

# Interpolasi
df['nilai'].interpolate()
```

Kapan pakai: Data pattern

Outlier Detection: IQR Method

```
# IQR (Interquartile Range) Method
Q1 = df['pagu'].quantile(0.25)
Q3 = df['pagu'].quantile(0.75)
IQR = Q3 - Q1

# Define bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Detect outliers
outliers = df[(df['pagu'] < lower_bound) | (df['pagu'] > upper_bound)]

print(f"Outliers: {len(outliers)} ({len(outliers)/len(df)*100:.2f}%)")

# Visualize dengan box plot
plt.figure(figsize=(12, 6))
df['pagu'].plot(kind='box', vert=False)
plt.title('Box Plot untuk Deteksi Outliers')
plt.show()
```

Outlier Detection: Z-Score Method

```python
from scipy import stats

# Calculate Z-scores
z_scores = np.abs(stats.zscore(df['pagu'].dropna()))

# Define threshold (typically 3)
threshold = 3

# Detect outliers
outlier_indices = np.where(z_scores > threshold)[0]
outliers_zscore = df.iloc[outlier_indices]

print(f"Outliers (Z-score): {len(outliers_zscore)}")

# Visualize
plt.figure(figsize=(12, 6))
plt.scatter(range(len(z_scores)), z_scores, alpha=0.5)
plt.axhline(y=threshold, color='r', linestyle='--', label=f'Threshold={threshold}')
plt.title('Z-Score Distribution')
plt.ylabel('Z-Score')
plt.legend()
plt.show()
```

🔧 Outlier Treatment

1. Remove Outliers

```
# Remove outliers
df_no_outliers = df[
    (df['pagu'] >= lower_bound) &
    (df['pagu'] <= upper_bound)
]
```

2. Winsorization (Capping)

```
# Cap at bounds
df['pagu_capped'] = df['pagu'].clip(
    lower=lower_bound,
    upper=upper_bound
)
```

3. Transformation

```
# Log perubahan
df['pagu_log'] = np.log1p(df['pagu'])

# Square root
df['pagu_sqrt'] = np.sqrt(df['pagu'])

# Box-Cox perubahan
from scipy.stats import boxcox
df['pagu_boxcox'], _ = boxcox(
    df['pagu'] + 1
)
```

🏷️ Encoding Categorical Variables

```python
from sklearn.persiapan data import LabelEncoder, OneHotEncoder

# 1. Label Encoding (Ordinal)
le = LabelEncoder()
df['metode_encoded'] = le.fit_transform(df['metode_pengadaan'])

# 2. One-Hot Encoding (Nominal)
df_encoded = pd.get_dummies(df, columns=['metode_pengadaan', 'jenis_pengadaan'])

# Alternative dengan sklearn
from sklearn.persiapan data import OneHotEncoder
ohe = OneHotEncoder(sparse=False, drop='first')
encoded = ohe.fit_transform(df[['metode_pengadaan']])

# 3. Frequency Encoding
freq_map = df['nama_satker'].value_counts().to_dict()
df['satker_freq'] = df['nama_satker'].map(freq_map)
```

⚙ Feature Engineering: Date/Time

```
# Extract date components
df['tgl'] = pd.to_datetime(df['tgl_pengumuman_paket'])

df['year'] = df['tgl'].dt.year
df['month'] = df['tgl'].dt.month
df['quarter'] = df['tgl'].dt.quarter
df['day'] = df['tgl'].dt.day
df['dayofweek'] = df['tgl'].dt.dayofweek
df['day_name'] = df['tgl'].dt.day_name()
df['week'] = df['tgl'].dt.isocalendar().week
df['is_weekend'] = df['dayofweek'].isin([5, 6]).astype(int)
df['is_month_start'] = df['tgl'].dt.is_month_start.astype(int)
df['is_month_end'] = df['tgl'].dt.is_month_end.astype(int)

# Time features
df['days_since_start'] = (df['tgl'] - df['tgl'].min()).dt.days
```

🔢 Feature Engineering: Binning

```
# Equal-width binning
df['pagu_binned'] = pd.cut(df['pagu'],
                           bins=5,
                           labels=['Very Rendah', 'Rendah', 'Sedang', 'Tinggi', 'Very Tinggi'])

# Custom bins
bins = [0, 100_000_000, 1_000_000_000, 10_000_000_000, float('inf')]
labels = ['Small', 'Sedang', 'Large', 'Very Large']
df['pagu_category'] = pd.cut(df['pagu'], bins=bins, labels=labels)

# Quantile-based binning (equal frequency)
df['pagu_quartile'] = pd.qcut(df['pagu'], q=4, labels=['Q1', 'Q2', 'Q3', 'Q4'])

# View distribution
print(df['pagu_category'].value_counts())
```

📝 Feature Engineering: Text Features

```python
# String length
df['nama_paket_length'] = df['nama_paket'].str.len()

# Word count
df['nama_paket_words'] = df['nama_paket'].str.split().str.len()

# Contains specific words
df['is_konstruksi'] = df['nama_paket'].str.contains('Konstruksi', case=False, na=False).astype(int)
df['is_pengadaan'] = df['nama_paket'].str.contains('Pengadaan', case=False, na=False).astype(int)

# Extract patterns dengan regex
import re
df['has_number'] = df['nama_paket'].str.contains(r'\d+', na=False).astype(int)

# Uppercase ratio
df['uppercase_ratio'] = df['nama_paket'].apply(
    lambda x: sum(1 for c in str(x) if c.isupper()) / len(str(x)) if x else 0
)
```

📊 Part 2: Analisis Data Berdasarkan Waktu

Penjelasan: Time series = data yang punya urutan waktu (harian, bulanan, dll)
Bisa lihat trend, pola musiman, prediksi masa depan

Notebook: `02_time_series.ipynb`

Time Series = Data dengan timestamp

Applications:

- Trend analysist
- Seasonality detection
- Forecasting
- Anomaly detection

📊 Part 2: Analisis Data Berdasarkan Waktu (lanjutan)

Dataset RUP:

- `tgl_pengumuman_paket`
- `tgl_awal_pemilihan`, `tgl_akhir_pemilihan`
- `tgl_awal_kontrak`, `tgl_akhir_kontrak`

DateTime Operations

```
# Convert to datetime
df['tgl'] = pd.to_datetime(df['tgl_pengumuman_paket'], errors='coerce')

# Set as index
df_ts = df.set_index('tgl').sort_index()

# Extract components
df_ts['year'] = df_ts.index.year
df_ts['month'] = df_ts.index.month
df_ts['day'] = df_ts.index.day

# Date arithmetic
df_ts['next_week'] = df_ts.index + pd.Timedelta(weeks=1)
df_ts['prev_month'] = df_ts.index - pd.DateOffset(months=1)

# Duration
df['durasi_kontrak'] = (df['tgl_akhir_kontrak'] - df['tgl_awal_kontrak']).dt.days
```

Resampling: Agregasi Time Series

```
# Daily merge
daily = df_ts.resample('D').size()

# Weekly merge
weekly = df_ts.resample('W').agg({
    'pagu': ['sum', 'mean', 'count']
})

# Monthly merge
monthly = df_ts.resample('M').agg({
    'pagu': 'sum',
    'kd_rup': 'count'
})

# Quarterly merge
quarterly = df_ts.resample('Q').agg({
    'pagu': ['sum', 'mean'],
    'kd_rup': 'count'
})

# Plot
monthly['pagu_sum'].plot(figsize=(12, 6), title='Monthly Total Pagu')
plt.show()
```

📈 Moving Averages

```
# Simple Moving Average (SMA)
daily_counts = df_ts.resample('D').size()

daily_counts_df = pd.DataFrame(daily_counts, columns=['count'])
daily_counts_df['SMA_7'] = daily_counts_df['count'].rolling(window=7).mean()
daily_counts_df['SMA_14'] = daily_counts_df['count'].rolling(window=14).mean()
daily_counts_df['SMA_30'] = daily_counts_df['count'].rolling(window=30).mean()

# Exponential Moving Average (EMA)
daily_counts_df['EMA_7'] = daily_counts_df['count'].ewm(span=7).mean()
daily_counts_df['EMA_14'] = daily_counts_df['count'].ewm(span=14).mean()

# Plot
daily_counts_df.plot(figsize=(14, 6))
plt.title('Pengumuman Paket dengan Moving Averages')
plt.ylabel('Jumlah Paket')
plt.legend()
plt.show()
```

Rolling Statistics

```
# Rolling window statistics
window = 7  # 7-day window

rolling_stats = pd.DataFrame({
    'count': daily_counts,
    'mean': daily_counts.rolling(window=window).mean(),
    'std': daily_counts.rolling(window=window).std(),
    'min': daily_counts.rolling(window=window).min(),
    'max': daily_counts.rolling(window=window).max()
})

# Plot with confidence interval
fig, ax = plt.subplots(figsize=(14, 6))
ax.plot(rolling_stats['count'], label='Actual', alpha=0.5)
ax.plot(rolling_stats['mean'], label=f'{window}-day MA', linewidth=2)
ax.fill_between(rolling_stats.index,
                rolling_stats['mean'] - rolling_stats['std'],
                rolling_stats['mean'] + rolling_stats['std'],
                alpha=0.2, label='±1 Std Dev')
plt.legend()
plt.show()
```

📈 Cumulative Calculations

```
# Cumulative sum (running total)
monthly = df_ts.resample('M').size()
monthly_cumsum = monthly.cumsum()

# Plot
fig, axes = plt.subplots(2, 1, figsize=(14, 10))

axes[0].bar(monthly.index, monthly.values)
axes[0].set_title('Monthly Paket Count')
axes[0].set_ylabel('Count')

axes[1].plot(monthly_cumsum.index, monthly_cumsum.values, marker='o')
axes[1].set_title('Cumulative Paket Count')
axes[1].set_ylabel('Cumulative Count')

plt.tight_layout()
plt.show()
```

Growth Rate Calculations

```
# Calculate growth rates
monthly = df_ts.resample('M').size()

# Day-over-day (DoD)
daily = df_ts.resample('D').size()
daily_growth = daily.pct_change() * 100

# Month-over-month (MoM)
monthly_growth = monthly.pct_change() * 100

# Year-over-year (YoY)
yearly = df_ts.resample('Y').size()
yoy_growth = yearly.pct_change() * 100

# Plot
monthly_growth.plot(kind='bar', figsize=(14, 6))
plt.title('Month-over-Month Growth Rate (%)')
plt.ylabel('Growth Rate (%)')
plt.axhline(y=0, color='r', linestyle='--')
plt.show()
```

🔄 Seasonality Analysis

```python
# Decompose time series
from statsmodels.tsa.seasonal import seasonal_decompose

# Prepare data (resample monthly)
monthly_data = df_ts.resample('M').size()

# Decomposition
decomposition = seasonal_decompose(monthly_data, model='additive', period=12)

# Plot components
fig, axes = plt.subplots(4, 1, figsize=(14, 12))

decomposition.observed.plot(ax=axes[0], title='Observed')
decomposition.trend.plot(ax=axes[1], title='Trend')
decomposition.seasonal.plot(ax=axes[2], title='Seasonal')
decomposition.resid.plot(ax=axes[3], title='Residual')

plt.tight_layout()
plt.show()
```

📊 Part 3: Analisis Statistik Dasar

Penjelasan: Pakai statistik untuk validasi asumsi dan ambil keputusan dari data
Contoh: Apakah perbedaan ini signifikan atau cuma kebetulan?

Notebook: `03_statistical_analysis.ipynb`

Statistical Analysis untuk:

- Paham distribusi data
- Test hypothesis
- Make data-driven decisions
- Validate assumptions

Part 3: Analisis Statistik Dasar (lanjutan)

Yang akan dipelajari:

- Descriptive statistics
- Correlation analysis
- Distribution analysis
- Hypothesis testing
- A/B testing

📈 Descriptive Statistics

```python
# Central tendency
mean_pagu = df['pagu'].mean()
median_pagu = df['pagu'].median()
mode_pagu = df['pagu'].mode()[0]

# Dispersion
variance = df['pagu'].var()
std_dev = df['pagu'].std()
range_val = df['pagu'].max() - df['pagu'].min()
iqr = df['pagu'].quantile(0.75) - df['pagu'].quantile(0.25)

# Coefisien of Variation
cv = (std_dev / mean_pagu) * 100

# Shape
from scipy.stats import skew, kurtosis
skewness = skew(df['pagu'].dropna())
kurt = kurtosis(df['pagu'].dropna())

print(f"Skewness: {skewness:.2f} ({'right' if skewness > 0 else 'left'} skewed)")
print(f"Kurtosis: {kurt:.2f}")
```

🔗 Correlation Analysis

```python
# Pearson correlation (linear relationship)
from scipy.stats import pearsonr

corr, p_value = pearsonr(df['pagu'], df['nilai_pdn_pekerjaan'])
print(f"Pearson r: {corr:.3f}, p-value: {p_value:.4f}")

# Spearman correlation (monotonic relationship)
from scipy.stats import spearmanr

corr_s, p_value_s = spearmanr(df['pagu'], df['nilai_pdn_pekerjaan'])
print(f"Spearman ρ: {corr_s:.3f}, p-value: {p_value_s:.4f}")

# Correlation matrix
numeric_cols = df.select_dtypes(include=[np.number]).columns
corr_matrix = df[numeric_cols].corr()

# Heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix')
plt.show()
```

Distribution Analysis

```python
# Test normality
from scipy.stats import shapiro, normaltest

# Shapiro-Wilk test
stat, p_value = shapiro(df['pagu'].sample(5000))  # Max 5000 samples
print(f"Shapiro-Wilk: statistic={stat:.4f}, p-value={p_value:.4f}")

if p_value > 0.05:
    print("Data appears normal (fail to reject H0)")
else:
    print("Data is not normal (reject H0)")

# D'Agostino-Pearson test
stat, p_value = normaltest(df['pagu'].dropna())
print(f"D'Agostino: statistic={stat:.4f}, p-value={p_value:.4f}")

# Q-Q plot
from scipy.stats import probplot
probplot(df['pagu'], dist="norm", plot=plt)
plt.title('Q-Q Plot')
plt.show()
```

🔬 Hypothesis Testing: T-Test

```python
from scipy.stats import ttest_ind

# Compare pagu between 2 metode
tender = df[df['metode_pengadaan'] == 'Tender']['pagu'].dropna()
langsung = df[df['metode_pengadaan'] == 'Penunjukan Langsung']['pagu'].dropna()

# Independent t-test
t_stat, p_value = ttest_ind(tender, langsung)

print(f"T-statistic: {t_stat:.4f}")
print(f"P-value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print(f"Reject H0: Means are significantly different (p={p_value:.4f} < {alpha})")
else:
    print(f"Fail to reject H0: No significant difference (p={p_value:.4f} >= {alpha})")
```

🔬 Hypothesis Testing: ANOVA

```python
from scipy.stats import f_oneway

# Compare pagu across multiple metode
groups = []
for metode in df['metode_pengadaan'].unique():
    groups.append(df[df['metode_pengadaan'] == metode]['pagu'].dropna())

# One-way ANOVA
f_stat, p_value = f_oneway(*groups)

print(f"F-statistic: {f_stat:.4f}")
print(f"P-value: {p_value:.4f}")

if p_value < 0.05:
    print("At least one group mean is different")
else:
    print("No significant difference between groups")
```

🔬 Chi-Square Test

```python
from scipy.stats import chi2_contingency

# Contingency table
contingency_table = pd.crosstab(df['metode_pengadaan'],
                                df['jenis_pengadaan'])

# Chi-square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

print(f"Chi-square: {chi2:.4f}")
print(f"P-value: {p_value:.4f}")
print(f"Degrees of freedom: {dof}")

if p_value < 0.05:
    print("Variables are dependent (reject H0)")
else:
    print("Variables are independent (fail to reject H0)")

# Visualize
sns.heatmap(contingency_table, annot=True, fmt='d', cmap='YlOrRd')
plt.title('Contingency Table: Metode vs Jenis')
plt.show()
```

🧪 A/B Testing Fundamentals

```
# Simulate A/B test
# Group A: Tender, Group B: Penunjukan Langsung

group_a = df[df['metode_pengadaan'] == 'Tender']['pagu'].dropna()
group_b = df[df['metode_pengadaan'] == 'Penunjukan Langsung']['pagu'].dropna()

# Calculate metrics
metrics = {
    'Group A': {
        'n': len(group_a),
        'mean': group_a.mean(),
        'std': group_a.std()
    },
    'Group B': {
        'n': len(group_b),
        'mean': group_b.mean(),
        'std': group_b.std()
    }
}
```

🧪 A/B Testing Fundamentals (lanjutan)

```
# Calculate lift
lift = ((metrics['Group B']['mean'] - metrics['Group A']['mean']) /
        metrics['Group A']['mean'] * 100)

print(f"Lift: {lift:.2f}%")
```

🧪 A/B Testing: Statistical Significance

```python
from scipy.stats import ttest_ind

# T-test
t_stat, p_value = ttest_ind(group_a, group_b)

# Calculate confidence interval
from scipy import stats
confidence = 0.95
ci = stats.t.interval(confidence, len(group_a) + len(group_b) - 2)

# Report results
print("=" * 60)
print("A/B TEST RESULTS")
print("=" * 60)
print(f"Group A (n={metrics['Group A']['n']}): Mean = {metrics['Group A']['mean']:,.0f}")
print(f"Group B (n={metrics['Group B']['n']}): Mean = {metrics['Group B']['mean']:,.0f}")
print(f"Lift: {lift:.2f}%")
print(f"T-statistic: {t_stat:.4f}")
print(f"P-value: {p_value:.4f}")
```

🧪 A/B Testing: Statistical Significance (lanjutan)

```
if p_value < 0.05:
    print("✅ Result is statistically significant!")
else:
    print("❌ Result is NOT statistically significant")
```

🎯 Latihan Praktis

Latihan

1. **Data Cleaning**
    - Tangani nilai yang hilang in your dataset
    - Detect and treat outliers
    - Create new features from existing columns
2. **Time Series**
    - Analyze monthly trends
    - Calculate growth rates
    - Detect seasonality

🎯 Latihan Praktis (lanjutan)

3. **Statistical Testing**
    - Compare mengelompokkan dengan t-test
    - Correlation analysis
    - Test for normality

💡 Praktik Terbaik

Data Cleaning

1. **Understand before action** 🔍
    - Investigasi KENAPA data is missing
    - Jangan asal remove outliers
2. **Dokumentasikan keputusan** 📝
    - Why you filled/dropped
    - Keep original data

💡 Praktik Terbaik (lanjutan)

3. **Validate results** ✅
    - Check distributions before/after
    - Periksa kewajaran
4. **Pertimbangkan pengetahuan domain** 🎓
    - Outliers bisa valid di konteks tertentu
    - Missing bisa punya arti

💡 Praktik Terbaik (lanjutan)

Statistical Analysis

1. **Check assumptions** ⚖️
    - Normalitas, independensi, homogenitas
2. **Choose right test** 🎯
    - Parametrik vs non-parametrik
    - Ukuran sampel penting
3. **Interpret correctly** 🤔
    - P-value ≠ ukuran efek
    - Statistical ≠ signifikansi praktis

💡 Praktik Terbaik (lanjutan)

4. **Multiple testing correction** 🔄
    - Bonferroni, FDR saat doing multiple tests

📚 Poin Penting

- ✅ Data cleaning adalah critical step (80% effort!)
- ✅ Tangani nilai yang hilang berdasarkan pattern & domain
- ✅ Outliers perlu investigation, bukan auto-remove
- ✅ Feature engineering boosts analysis quality
- ✅ Time series analysis reveals trends & patterns
- ✅ Statistical tests validate hypotheses
- ✅ Always check assumptions before testing
- ✅ P-value < 0.05 doesn't mean signifikansi praktis

Clean data = Better insights! 🧹✨

🔗 Resources

- **Pandas:** https://pandas.pydata.org/docs/
- **SciPy Stats:** https://docs.scipy.org/doc/scipy/reference/stats.html
- **Statsmodels:** https://www.statsmodels.org/
- **Practical Statistics for Data Scientists** (Book)

Sesi Selanjutnya

Sesi 5: Dashboard Interaktif dengan Streamlit

- Building web apps with Streamlit
- Interactive components
- Production-ready dashboards

BREAK sampai 13:00 🍽️

🎉 Selesai Sesi 4!

Great Progress! 👏

Makan siang sampai 13:00

Pertanyaan?? 🙋

Selanjutnya: Building Dashboards! 🚀

📞 Tetap Terhubung

Resources & Support

- 👤 **Nama:** [Kurnia Ramadhan,ST.,M.Eng]
- 📧 **Email:** [kurnia@ramadhan.me]

We're here to support your journey! 🤝