📋Clase 33 - Pandas KPI’s

Creación de KPIs con Pandas

📊 Introducción

Los **KPIs** (Key Performance Indicators) son métricas cuantificables que permiten evaluar el rendimiento de un negocio, proyecto o proceso en relación con objetivos estratégicos. En Data Science, Pandas es la herramienta fundamental para calcular, analizar y monitorear estos indicadores.

🎯 ¿Qué es un KPI?

Un KPI es una medida del rendimiento que:

- **Es cuantificable**: Se puede medir con números
- **Es accionable**: Permite tomar decisiones
- **Es relevante**: Está alineado con objetivos de negocio
- **Es temporal**: Se mide en períodos específicos
- **Es comparable**: Permite comparar contra objetivos o períodos anteriores

Características de un buen KPI

- **Específico**: Mide un aspecto concreto del negocio
- **Medible**: Tiene unidades claras (%, $, cantidad, tiempo)
- **Alcanzable**: Es realista
- **Relevante**: Impacta en objetivos estratégicos
- **Temporal**: Tiene un marco de tiempo definido

Tipos de KPIs

- **KPIs de Rendimiento**: Miden eficiencia (ej: tasa de conversión)
- **KPIs de Calidad**: Miden calidad del producto/servicio (ej: tasa de error)
- **KPIs Financieros**: Miden salud financiera (ej: ROI, margen de ganancia)
- **KPIs de Crecimiento**: Miden expansión (ej: tasa de crecimiento mensual)
- **KPIs Operacionales**: Miden procesos internos (ej: tiempo de respuesta)

📚 Conceptos de Pandas Necesarios

Antes de calcular KPIs, necesitas conocer:

- `groupby()`: Agrupar datos por categorías
- `agg()`: Aplicar múltiples funciones de agregación
- `resample()`: Agrupar por períodos de tiempo
- `rolling()`: Calcular métricas móviles
- `pct_change()`: Calcular cambios porcentuales
- `merge()`: Combinar diferentes fuentes de datos
- `pivot_table()`: Crear tablas dinámicas

🟢 Ejemplos

Ejemplo 1: Tasa de Conversión

> **Definición**: Porcentaje de visitantes que completan una acción deseada.

```
import pandas as pd
import numpy as np

# Crear datos de ejemplo
data = {
```

```
    'fecha': pd.date_range('2024-01-01', periods=30, freq='D'),
    'visitantes': np.random.randint(100, 500, 30),
    'conversiones': np.random.randint(10, 50, 30)
}

df = pd.DataFrame(data)

# Calcular KPI: Tasa de Conversión
df['tasa_conversion'] = (df['conversiones'] / df['visitantes']) * 100

# Ver resultados
print(df[['fecha', 'visitantes', 'conversiones', 'tasa_conversion']].head(10))
print(f"\nTasa de conversión promedio: {df['tasa_conversion'].mean():.2f}%")
```

Ejemplo 2: Ticket Promedio

> **Definición**: Valor promedio de cada transacción.

```
# Datos de ventas
ventas_data = {
    'transaccion_id': range(1, 101),
    'monto': np.random.uniform(20, 500, 100),
    'fecha': pd.date_range('2024-01-01', periods=100, freq='H')
}

df_ventas = pd.DataFrame(ventas_data)

# KPI: Ticket Promedio
ticket_promedio = df_ventas['monto'].mean()

print(f"Ticket Promedio: ${ticket_promedio:.2f}")

# Ticket Promedio por día
df_ventas['fecha_dia'] = df_ventas['fecha'].dt.date
ticket_diario = df_ventas.groupby('fecha_dia')['monto'].mean()

print("\nTicket Promedio Diario:")
print(ticket_diario)
```

Ejemplo 3: Tasa de Retención de Clientes

> **Definición**: Porcentaje de clientes que regresan en un período.

```
# Datos de clientes
clientes_data = {
    'cliente_id': [1, 2, 3, 1, 4, 2, 5, 1, 3, 6],
    'fecha_compra': pd.to_datetime([
        '2024-01-05', '2024-01-06', '2024-01-07', '2024-02-10',
        '2024-02-11', '2024-02-12', '2024-02-15', '2024-03-05',
        '2024-03-08', '2024-03-10'
    ])
}

df_clientes = pd.DataFrame(clientes_data)

# Agregar mes
df_clientes['mes'] = df_clientes['fecha_compra'].dt.to_period('M')

# Clientes únicos por mes
```

```
clientes_por_mes = df_clientes.groupby('mes')['cliente_id'].nunique()

print("Clientes únicos por mes:")
print(clientes_por_mes)

# Clientes que compraron en meses consecutivos
clientes_enero = set(df_clientes[df_clientes['mes'] == '2024-01']['cliente_id'])
clientes_febrero = set(df_clientes[df_clientes['mes'] == '2024-02']['cliente_id'])

clientes_retenidos = clientes_enero & clientes_febrero
tasa_retencion = (len(clientes_retenidos) / len(clientes_enero)) * 100

print(f"\nTasa de Retención Enero→Febrero: {tasa_retencion:.2f}%")
```

Ejemplo 4: CAC (Customer Acquisition Cost)

> **Definición**: Costo promedio de adquirir un nuevo cliente.

```
# Datos de marketing y ventas
marketing_data = {
    'mes': ['2024-01', '2024-02', '2024-03', '2024-04'],
    'gasto_marketing': [5000, 6000, 5500, 7000],
    'nuevos_clientes': [50, 60, 55, 70]
}

df_marketing = pd.DataFrame(marketing_data)

# KPI: CAC
df_marketing['CAC'] = df_marketing['gasto_marketing'] / df_marketing['nuevos_clientes']

print(df_marketing)
print(f"\nCAC Promedio: ${df_marketing['CAC'].mean():.2f}")

# Tendencia del CAC
df_marketing['variacion_CAC'] = df_marketing['CAC'].pct_change() * 100

print("\nVariación mensual del CAC:")
print(df_marketing[['mes', 'CAC', 'variacion_CAC']])
```

Ejemplo 5: Churn Rate (Tasa de Cancelación)

> **Definición**: Porcentaje de clientes que dejan de usar el servicio.

```
# Datos de suscripciones
suscripciones = {
    'mes': pd.date_range('2024-01', periods=6, freq='M'),
    'clientes_inicio': [1000, 1050, 1080, 1060, 1090, 1100],
    'clientes_nuevos': [100, 80, 50, 70, 60, 55],
    'clientes_cancelados': [50, 50, 70, 40, 50, 45]
}

df_subs = pd.DataFrame(suscripciones)

# Calcular clientes al final del mes
df_subs['clientes_fin'] = (df_subs['clientes_inicio'] +
                           df_subs['clientes_nuevos'] -
                           df_subs['clientes_cancelados'])

# KPI: Churn Rate
```

```
df_subs['churn_rate'] = (df_subs['clientes_cancelados'] /
                         df_subs['clientes_inicio']) * 100

# KPI: Growth Rate
df_subs['growth_rate'] = ((df_subs['clientes_fin'] - df_subs['clientes_inicio']) /
                          df_subs['clientes_inicio']) * 100

print(df_subs[['mes', 'churn_rate', 'growth_rate']])
print(f"\nChurn Rate Promedio: {df_subs['churn_rate'].mean():.2f}%")
```

Ejemplo 6: KPIs de Ventas Multiproducto

> **Definición**: Análisis de ventas por producto y categoría.

```
# Datos de ventas detalladas
np.random.seed(42)
ventas_detalladas = {
    'fecha': pd.date_range('2024-01-01', periods=200, freq='D').repeat(3)[:500],
    'producto': np.random.choice(['Laptop', 'Mouse', 'Teclado', 'Monitor', 'Auriculares'], 500),
    'categoria': np.random.choice(['Computadoras', 'Accesorios'], 500),
    'cantidad': np.random.randint(1, 10, 500),
    'precio_unitario': np.random.uniform(10, 1000, 500)
}

df_ventas_det = pd.DataFrame(ventas_detalladas)
df_ventas_det['ingreso_total'] = df_ventas_det['cantidad'] * df_ventas_det['precio_unitario']

# KPI 1: Ingreso por Producto
ingreso_por_producto = df_ventas_det.groupby('producto').agg({
    'ingreso_total': 'sum',
    'cantidad': 'sum'
}).sort_values('ingreso_total', ascending=False)

print("Ingreso por Producto:")
print(ingreso_por_producto)

# KPI 2: Participación de Mercado por Producto
ingreso_por_producto['market_share'] = (
    ingreso_por_producto['ingreso_total'] /
    ingreso_por_producto['ingreso_total'].sum() * 100
)

print("\nMarket Share por Producto:")
print(ingreso_por_producto['market_share'])

# KPI 3: Ingreso Promedio Diario
df_ventas_det['fecha_dia'] = df_ventas_det['fecha'].dt.date
ingreso_diario = df_ventas_det.groupby('fecha_dia')['ingreso_total'].sum()

print(f"\nIngreso Promedio Diario: ${ingreso_diario.mean():.2f}")
```

Ejemplo 7: Cohort Analysis y LTV (Lifetime Value)

> **Definición**: Análisis de cohortes para calcular el valor de vida del cliente.

```
# Generar datos de cohortes
np.random.seed(42)
n_clientes = 1000
```

```
cohort_data = []
for cliente_id in range(1, n_clientes + 1):
    fecha_primera_compra = pd.Timestamp('2023-01-01') + pd.Timedelta(days=np.random.randint(0, 365))
    cohort_mes = fecha_primera_compra.to_period('M')

    # Simular compras posteriores (probabilidad decreciente)
    n_compras = np.random.poisson(3)

    for i in range(n_compras):
        dias_desde_primera = np.random.randint(0, 365)
        fecha_compra = fecha_primera_compra + pd.Timedelta(days=dias_desde_primera)
        monto = np.random.uniform(20, 200)

        cohort_data.append({
            'cliente_id': cliente_id,
            'fecha_primera_compra': fecha_primera_compra,
            'cohort_mes': cohort_mes,
            'fecha_compra': fecha_compra,
            'monto': monto
        })

df_cohort = pd.DataFrame(cohort_data)
df_cohort['mes_compra'] = df_cohort['fecha_compra'].dt.to_period('M')

# Calcular meses desde la primera compra
df_cohort['meses_desde_primera'] = (
    (df_cohort['mes_compra'] - df_cohort['cohort_mes']).apply(lambda x: x.n)
)

# KPI 1: LTV por Cohorte
ltv_por_cohorte = df_cohort.groupby('cohort_mes').agg({
    'cliente_id': 'nunique',
    'monto': 'sum'
})
ltv_por_cohorte['LTV_promedio'] = (
    ltv_por_cohorte['monto'] / ltv_por_cohorte['cliente_id']
)

print("LTV por Cohorte:")
print(ltv_por_cohorte[['cliente_id', 'LTV_promedio']].head(10))

# KPI 2: Retención por Cohorte
retention_table = df_cohort.pivot_table(
    values='cliente_id',
    index='cohort_mes',
    columns='meses_desde_primera',
    aggfunc='nunique'
)

# Calcular porcentaje de retención
retention_pct = retention_table.div(retention_table[0], axis=0) * 100

print("\nTasa de Retención por Mes (primeras 5 cohortes, primeros 6 meses):")
print(retention_pct.iloc[:5, :6])
```

Ejemplo 8: KPIs con Rolling Windows y Tendencias

> **Definición**: Métricas móviles para identificar tendencias.

```
# Datos de ventas diarias
np.random.seed(42)
fechas = pd.date_range('2023-01-01', '2024-12-31', freq='D')
ventas_diarias = {
    'fecha': fechas,
    'ventas': np.random.poisson(100, len(fechas)) + np.sin(np.arange(len(fechas)) * 2 * np.pi / 365) * 20,
    'visitantes': np.random.poisson(1000, len(fechas))
}

df_trend = pd.DataFrame(ventas_diarias)
df_trend['conversion_rate'] = (df_trend['ventas'] / df_trend['visitantes']) * 100

# KPI 1: Media Móvil de 7 días
df_trend['ventas_ma7'] = df_trend['ventas'].rolling(window=7).mean()

# KPI 2: Media Móvil de 30 días
df_trend['ventas_ma30'] = df_trend['ventas'].rolling(window=30).mean()

# KPI 3: Crecimiento Inter-anual (YoY)
df_trend['ventas_yoy'] = df_trend['ventas'].pct_change(periods=365) * 100

# KPI 4: Volatilidad (desviación estándar móvil)
df_trend['volatilidad_7d'] = df_trend['ventas'].rolling(window=7).std()

# KPI 5: Tendencia (comparar MA7 vs MA30)
df_trend['tendencia'] = np.where(
    df_trend['ventas_ma7'] > df_trend['ventas_ma30'],
    'Alcista',
    'Bajista'
)

print("KPIs de Tendencia (últimos 10 días):")
print(df_trend[['fecha', 'ventas', 'ventas_ma7', 'ventas_ma30', 'tendencia']].tail(10))

# Análisis de estacionalidad
df_trend['mes'] = df_trend['fecha'].dt.month
estacionalidad = df_trend.groupby('mes')['ventas'].mean()

print("\nEstacionalidad por Mes:")
print(estacionalidad)
```

Ejemplo 9: Dashboard de KPIs Múltiples con Agregaciones Complejas

> **Definición**: Panel completo de KPIs empresariales.

```
# Simular datos empresariales completos
np.random.seed(42)
n_registros = 5000

empresa_data = {
    'fecha': pd.date_range('2023-01-01', periods=n_registros, freq='H'),
    'producto': np.random.choice(['Premium', 'Standard', 'Basic'], n_registros),
    'canal': np.random.choice(['Web', 'Móvil', 'Tienda Física'], n_registros),
    'region': np.random.choice(['Norte', 'Sur', 'Este', 'Oeste'], n_registros),
    'ventas': np.random.uniform(50, 500, n_registros),
    'costo': np.random.uniform(20, 300, n_registros),
    'tiempo_atencion_min': np.random.uniform(5, 60, n_registros)
}
```

```
df_empresa = pd.DataFrame(empresa_data)
df_empresa['ganancia'] = df_empresa['ventas'] - df_empresa['costo']
df_empresa['margen'] = (df_empresa['ganancia'] / df_empresa['ventas']) * 100

# Crear Dashboard de KPIs
def crear_dashboard_kpis(df, fecha_inicio=None, fecha_fin=None):
    """
    Genera un dashboard completo de KPIs
    """
    # Filtrar por fechas si se especifica
    if fecha_inicio:
        df = df[df['fecha'] >= fecha_inicio]
    if fecha_fin:
        df = df[df['fecha'] <= fecha_fin]

    dashboard = {}

    # KPIs Financieros
    dashboard['ingresos_totales'] = df['ventas'].sum()
    dashboard['ganancia_total'] = df['ganancia'].sum()
    dashboard['margen_promedio'] = df['margen'].mean()
    dashboard['roi'] = (df['ganancia'].sum() / df['costo'].sum()) * 100

    # KPIs Operacionales
    dashboard['transacciones_totales'] = len(df)
    dashboard['ticket_promedio'] = df['ventas'].mean()
    dashboard['tiempo_atencion_promedio'] = df['tiempo_atencion_min'].mean()

    # KPIs por Dimensiones
    dashboard['ventas_por_producto'] = df.groupby('producto').agg({
        'ventas': 'sum',
        'ganancia': 'sum',
        'margen': 'mean'
    }).to_dict()

    dashboard['ventas_por_canal'] = df.groupby('canal')['ventas'].sum().to_dict()

    dashboard['ventas_por_region'] = df.groupby('region')['ventas'].sum().to_dict()

    # KPI de Concentración (Top productos)
    top_productos = df.groupby('producto')['ventas'].sum().sort_values(ascending=False)
    dashboard['concentracion_top3'] = (
        top_productos.head(3).sum() / top_productos.sum() * 100
    )

    return dashboard

# Generar dashboard mensual
df_empresa['mes'] = df_empresa['fecha'].dt.to_period('M')

print("=== DASHBOARD DE KPIs ===\n")

# KPIs Globales
dashboard_global = crear_dashboard_kpis(df_empresa)

print("KPIs FINANCIEROS:")
print(f"  Ingresos Totales: ${dashboard_global['ingresos_totales']:,.2f}")
print(f"  Ganancia Total: ${dashboard_global['ganancia_total']:,.2f}")
print(f"  Margen Promedio: {dashboard_global['margen_promedio']:.2f}%")
```

```
print(f"  ROI: {dashboard_global['roi']:.2f}%")

print("\nKPIs OPERACIONALES:")
print(f"  Transacciones: {dashboard_global['transacciones_totales']:,}")
print(f"  Ticket Promedio: ${dashboard_global['ticket_promedio']:.2f}")
print(f"  Tiempo Atención: {dashboard_global['tiempo_atencion_promedio']:.2f} min")

print("\nVENTAS POR PRODUCTO:")
for producto, metrics in dashboard_global['ventas_por_producto'].items():
    print(f"  {producto}: ${metrics['ventas']:,.2f}")

print("\nVENTAS POR CANAL:")
for canal, ventas in dashboard_global['ventas_por_canal'].items():
    print(f"  {canal}: ${ventas:,.2f}")

print(f"\nConcentración Top 3: {dashboard_global['concentracion_top3']:.2f}%")

# Comparación mensual
kpis_mensuales = df_empresa.groupby('mes').agg({
    'ventas': 'sum',
    'ganancia': 'sum',
    'margen': 'mean'
}).reset_index()

kpis_mensuales['crecimiento_ventas'] = kpis_mensuales['ventas'].pct_change() * 100

print("\n=== EVOLUCIÓN MENSUAL ===")
print(kpis_mensuales.tail(6))
```

Ejemplo 10: KPIs Predictivos con Forecasting Simple

> **Definición**: Proyección de KPIs futuros basados en tendencias históricas.

```
# Datos históricos de ventas
np.random.seed(42)
fechas_hist = pd.date_range('2022-01-01', '2024-10-31', freq='M')
ventas_hist = 10000 + np.arange(len(fechas_hist)) * 500 + np.random.normal(0, 1000, len(fechas_hist))

df_hist = pd.DataFrame({
    'mes': fechas_hist,
    'ventas_reales': ventas_hist
})

# KPI 1: Tendencia lineal
from sklearn.linear_model import LinearRegression

X = np.arange(len(df_hist)).reshape(-1, 1)
y = df_hist['ventas_reales'].values

model = LinearRegression()
model.fit(X, y)

df_hist['ventas_tendencia'] = model.predict(X)

# Proyectar próximos 6 meses
meses_futuros = 6
X_futuro = np.arange(len(df_hist), len(df_hist) + meses_futuros).reshape(-1, 1)
ventas_proyectadas = model.predict(X_futuro)
```

```
df_futuro = pd.DataFrame({
    'mes': pd.date_range(df_hist['mes'].iloc[-1] + pd.DateOffset(months=1),
                         periods=meses_futuros, freq='M'),
    'ventas_proyectadas': ventas_proyectadas
})

# KPI 2: Tasa de crecimiento promedio
df_hist['crecimiento'] = df_hist['ventas_reales'].pct_change() * 100
tasa_crecimiento_promedio = df_hist['crecimiento'].mean()

print("=== KPIs PREDICTIVOS ===\n")
print(f"Tasa de Crecimiento Promedio Mensual: {tasa_crecimiento_promedio:.2f}%")
print(f"Pendiente de Crecimiento: ${model.coef_[0]:.2f}/mes")

print("\nPROYECCIÓN PRÓXIMOS 6 MESES:")
print(df_futuro)

# KPI 3: Calcular objetivo basado en crecimiento deseado
objetivo_crecimiento = 5  # 5% mensual
ventas_ultima = df_hist['ventas_reales'].iloc[-1]

objetivos = []
for i in range(1, meses_futuros + 1):
    objetivo = ventas_ultima * ((1 + objetivo_crecimiento/100) ** i)
    objetivos.append(objetivo)

df_futuro['objetivo_5pct'] = objetivos
df_futuro['gap_vs_objetivo'] = df_futuro['ventas_proyectadas'] - df_futuro['objetivo_5pct']

print("\nCOMPARACIÓN VS OBJETIVO:")
print(df_futuro[['mes', 'ventas_proyectadas', 'objetivo_5pct', 'gap_vs_objetivo']])

# KPI 4: Probabilidad de alcanzar objetivo
# Basado en desviación estándar histórica
std_hist = df_hist['ventas_reales'].std()
print(f"\nDesviación Estándar Histórica: ${std_hist:.2f}")
```

🎓 Mejores Prácticas para KPIs con Pandas

1. Organización de Datos

```
# ✅ BUENA PRÁCTICA: Usar tipos de datos apropiados
df['fecha'] = pd.to_datetime(df['fecha'])
df['categoria'] = df['categoria'].astype('category')
df['monto'] = df['monto'].astype('float64')

# ✅ BUENA PRÁCTICA: Índices apropiados para series temporales
df.set_index('fecha', inplace=True)
```

2. Funciones Reutilizables

```
def calcular_kpi_conversion(df, col_eventos, col_usuarios):
    """
    Calcula tasa de conversión

    Args:
        df: DataFrame con los datos
        col_eventos: Nombre de la columna de eventos exitosos
```

```
    col_usuarios: Nombre de la columna de usuarios totales

  Returns:
    Series con la tasa de conversión
  """
  return (df[col_eventos] / df[col_usuarios]) * 100

def calcular_crecimiento(df, columna, periodos=1):
  """
  Calcula tasa de crecimiento

  Args:
    df: DataFrame
    columna: Columna a analizar
    periodos: Número de períodos para comparar

  Returns:
    Series con el crecimiento porcentual
  """
  return df[columna].pct_change(periods=periodos) * 100
```

3. Validación de Datos

```
# ✅ BUENA PRÁCTICA: Validar antes de calcular KPIs
def validar_datos_kpi(df, columnas_requeridas):
  """Valida que el DataFrame tenga las columnas necesarias"""
  faltantes = set(columnas_requeridas) - set(df.columns)
  if faltantes:
    raise ValueError(f"Columnas faltantes: {faltantes}")

  # Verificar valores nulos
  nulos = df[columnas_requeridas].isnull().sum()
  if nulos.any():
    print(f"Advertencia: Valores nulos encontrados:\n{nulos[nulos > 0]}")

  return True
```

4. Documentación de KPIs

```
# ✅ BUENA PRÁCTICA: Documentar fórmulas y supuestos
KPIS_DEFINICIONES = {
  'CAC': {
    'nombre': 'Customer Acquisition Cost',
    'formula': 'Gasto Marketing / Nuevos Clientes',
    'unidad': 'USD',
    'objetivo': '< 100',
    'frecuencia': 'Mensual'
  },
  'LTV': {
    'nombre': 'Lifetime Value',
    'formula': 'Ingreso Promedio por Cliente * Vida Promedio Cliente',
    'unidad': 'USD',
    'objetivo': '> 500',
    'frecuencia': 'Trimestral'
  }
}
```